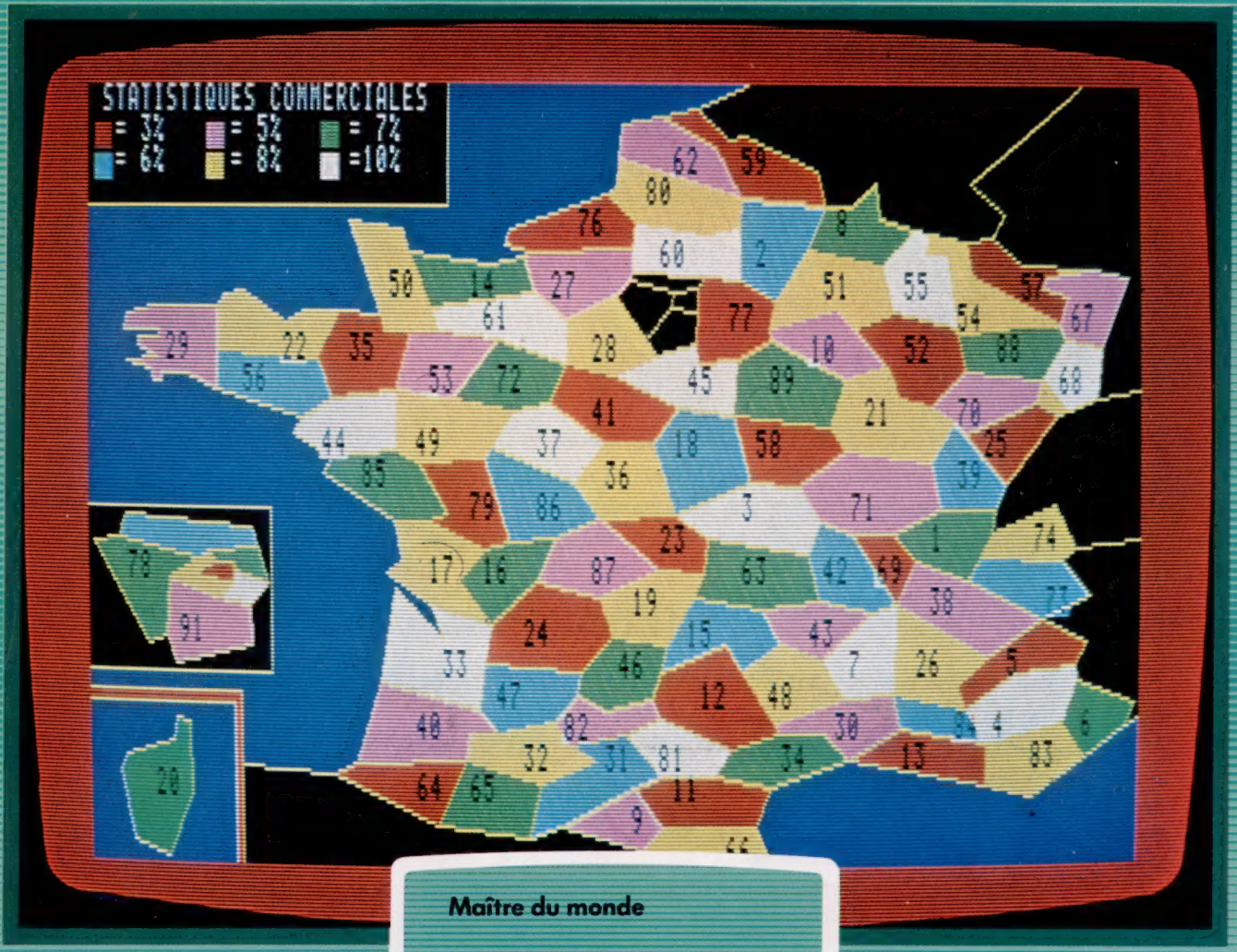
ABC INFORMATIQUE

N°58

COURS D'INFORMATIQUE PRATIQUE ET FAMILIALE



Maître du monde

Réveille-matin

Toshiba HX-10

Du micro à la compo

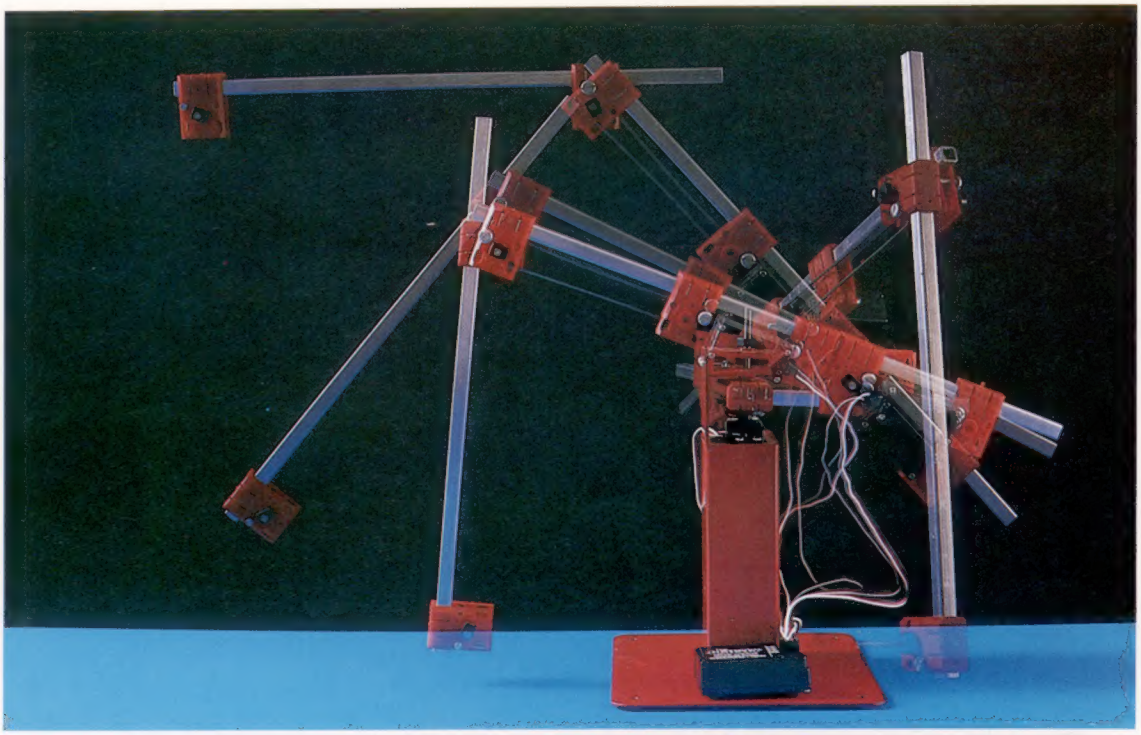
EDITIONS ATLAS

M6062-58-12-F 85FB-3,80FS-$1.95

Page manquante
(publicité et colophon)

Bras de fer

Après avoir vu les diverses méthodes utilisées pour contrôler le déplacement des robots, arrêtons-nous sur le contrôle et les mouvements possibles de leur « bras » et de leur « main ».



Double articulation
Le robot type des dix prochaines années sera un simple bras doté de toute une gamme de « mains » destinées à des usages spécifiques : industrie, ménage et bricolage. Très peu d'applications nécessitent en effet la machine autonome et pensante de la science-fiction. En revanche, une main-pince programmable et semi-intelligente constitue un appareil aussi important que la charrue ou le télescope. (Cl. Ian McKinnell.)

Le degré d'efficacité d'un robot dépend, en grande partie, de la précision de ses mouvements. De nombreux robots sont utilisés avant tout pour des opérations du type « prendre et mettre en place ». C'est pourquoi la conception du bras du robot est d'une importance primordiale.

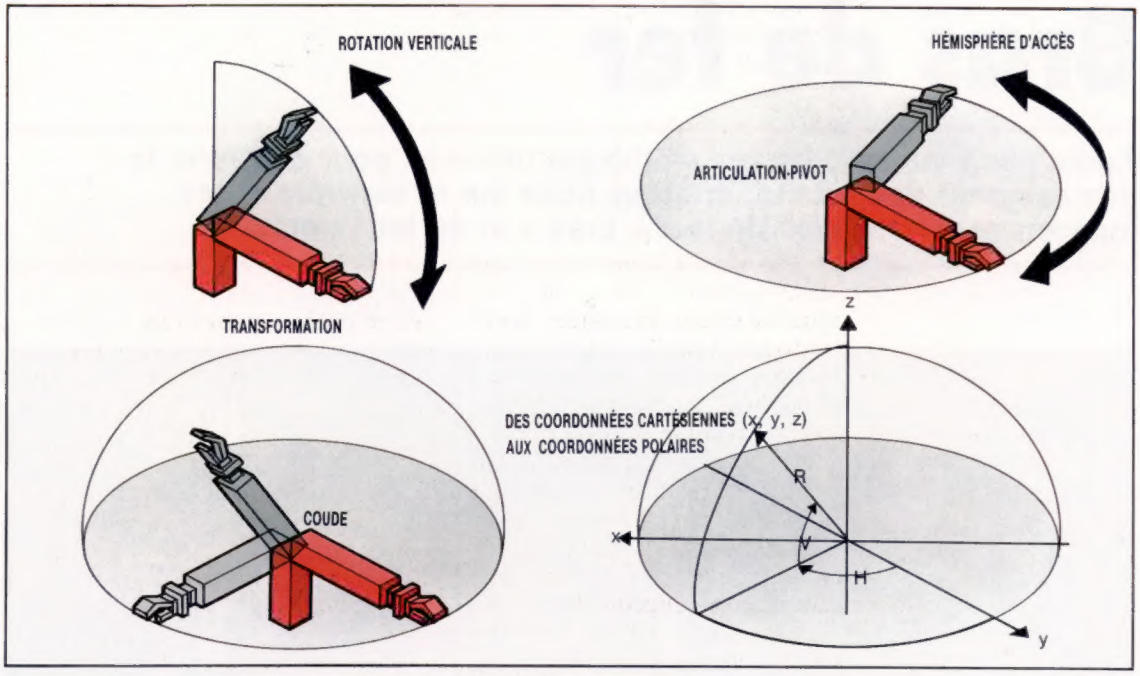
De manière générale, trois impératifs doivent être satisfaits. Il faut créer un système pour décrire à tout moment la position du bras, lequel doit comporter une « ossature » et comprendre un « système musculaire », afin d'être animé et dirigé. Les interactions entre ces composantes déterminent son aspect général.

Nous avons déjà décrit le système de coordonnées cartésiennes. Selon cette méthode, la position du robot sur le sol est repérée selon deux axes x et y se coupant à angle droit. Le même principe peut être appliqué à un bras de robot, mais nous avons besoin d'une troisième variable, car le bras se déplace dans un espace à trois dimensions. La variable z donnera la position verticale du bras. En utilisant ces trois variables x, y et z, on peut décrire la position du bras en tout point de l'espace.

Il est possible de construire un bras de robot qui se déplace exactement selon ces trois axes. Le résultat est quelque chose qui ressemble à un portique ou à une grue, et qui peut se déplacer verticalement, horizontalement et en profondeur (ou en combinant ces trois axes). Les bras de ce type sont bien adaptés aux travaux sur une zone qui ne varie pas. Par exemple, le robot peut être affecté à un plan où s'effectuent toutes ses tâches : un bras « cartésien » est alors tout à fait approprié. Mais cette méthode présente l'inconvénient d'être inutilisable pour des applications intervenant en dehors d'un plan de travail.

Une autre méthode pour décrire la position d'un bras de robot est d'utiliser des coordonnées cylindriques. Pour vous représenter ce dont il s'agit, pensez à une boîte de conserve vide : chaque point à l'intérieur de la boîte peut être décrit selon sa distance à l'axe (variable-distance r), sa distance sur le pourtour de la boîte par rapport à un point fixe (variable angulaire θ), et sa hauteur sur la boîte (autre variable-distance, z). L'utilisation de coordonnées cylindriques permet donc de mettre en œuvre un système de préhension d'objets situés à l'intérieur d'un cylindre imaginaire.

Les bras utilisant des coordonnées sphériques développent ce principe en précisant les positions en fonction de deux angles et d'une distance. Cette dernière est la longueur du bras, les angles



Rotations robotiques
Le bras le plus élémentaire, constitué d'une pince et d'une « articulation-coude » sur deux axes, est capable d'un positionnement précis dans un grand espace. Le coude comporte un pivot qui permet un déplacement vertical sur un demi-arc de cercle, et un autre qui autorise un mouvement circulaire plan sur un arc de cercle complet. Le bras se positionne, sur n'importe quel point situé dans l'hémisphère ainsi défini, par des rotations appropriées de ses pivots. Ces dernières se calculent par des opérations trigonométriques, à partir de coordonnées cartésiennes du point visé, de la manière suivante : H, la rotation du pivot horizontal, est égale à ARCTAN(x/y) ; V, la rotation verticale, à ARCSIN(z/R). On programme les positions cartésiennes des objets. Le bras ainsi programmé transforme chaque jeu de coordonnées en deux degrés de rotation, vertical et horizontal, transmis aux servomoteurs, qui effectuent le déplacement voulu. (Cl. Kevin Jones.)

étant les angles de rotation horizontale et verticale. Les coordonnées sphériques correspondent aux variables r, Θ et Φ. Pour l'ingénieur qui conçoit le robot, il est assez facile de mettre au point un bras télescopique, manœuvrable hydrauliquement par exemple. La dernière méthode de description de la position du bras dans l'espace (la plus courante) utilise des coordonnées de révolution. Ce système est conçu spécifiquement pour apparenter le contrôle du bras du robot à celui du bras de l'homme. Trois variables sont ici également nécessaires, mais toutes sont des angles : Θ, Φ et Υ. La première, Θ, théta, se réfère à l'angle de rotation de la base; la seconde, Φ, phi, renvoie à l'angle d'élévation du bras; la dernière, Υ, gamma, donne l'angle pour la deuxième articulation du bras.

Faire du muscle

Le système de coordonnées retenu dicte le type de l'ossature du bras du robot. Il reste à lui donner du muscle, afin de l'animer. L'énergie est en général de trois ordres : électrique, hydraulique ou pneumatique. Voyons en détail ces trois formes d'énergie.

Nous avons déjà abordé l'alimentation électrique en rapport avec le déplacement du robot. Les mêmes moteurs électriques pas à pas peuvent alimenter le bras du robot. Ils peuvent agir soit directement, en étant placés aux articulations (ils sont alors puissants et impriment une faible rotation), soit indirectement, par l'intermédiaire d'embrayages, poulies ou leviers.

Cependant, il serait nettement préférable de pousser plus loin l'analogie avec nos propres bras, en dotant le robot d'une énergie quasi « musculaire » de contraction et de détente. Cela s'obtient par une suite de pistons à chaque articulation, soit hydrauliques (à l'aide d'un liquide), soit pneumatiques (par de l'air sous pression). Pour de gros robots industriels, l'énergie pneumatique est préférable, dans la mesure où elle est capable de fournir des pressions élevées (donnant plus de force au bras), et parce que les fluides ne se compriment et ne se dilatent pas dans la même mesure que l'air.

Lorsqu'un piston est actionné par pression hydraulique dans un cylindre, il ne bondit pas et, en outre, il s'arrête précisément au point voulu. L'air comprimé ne permet pas un positionnement aussi précis. Quel que soit le système de pression utilisé, les pistons peuvent être à une ou à deux actions, pour produire le mouvement du bras. Ce type de puissance motrice est appelé « agent moteur linéaire ».

Il est possible d'apporter d'autres améliorations à ce système. Au lieu d'utiliser des pistons qui se contractent et se détendent dans un cylindre, et qui transmettent ensuite à l'articulation ce mouvement sous la forme d'une rotation, on peut mettre en œuvre un « agent moteur rotatif » qui produit directement la rotation par l'intermédiaire d'une pale dans un logement circulaire. Il s'agit d'un processus similaire à celui d'un moteur électrique pas à pas, mais la pression hydraulique autorise davantage de puissance. La pression pneumatique ne convient pas à ce genre d'applications.

Une fois le mécanisme du bras du robot adopté, il faut encore une « main » (ou extrémité), afin que le bras correctement positionné puisse réaliser une tâche déterminée. C'est le moment de penser au fonctionnement de la main humaine. Prenez le poignet : sa mobilité autorise de nombreux mouvements de la main. Lorsque vous tapez sur un clavier, par exemple, votre poignet vous permet d'effectuer des mouvements verticaux et vous évite d'avoir à manœuvrer l'avant-bras.

Le poignet autorise également un déplacement latéral de la main, qui permet l'accès à des tou-

ches éloignées. Sans cette caractéristique, il vous faudrait encore faire jouer l'articulation du coude. Le poignet permet enfin un mouvement d'enroulement sur lui-même, grâce auquel vous pouvez reposer votre main sur un clavier. Le pouce est alors tourné vers le haut et non plus vers le bas. Ce mouvement est le plus complexe et vous évite toute une gesticulation au niveau de l'articulation de l'épaule.

De façon idéale, donc, le robot devrait comporter un poignet sur le modèle de celui de l'homme et capable en outre de ces trois mouvements. Ces derniers — mouvements vertical, latéral et d'enroulement — se font selon deux directions : respectivement haut/bas, gauche/droite et « sens des aiguilles d'une montre/sens contraire ». Chacun correspond à ce que l'on appelle un degré de liberté. Le robot parfait comporterait donc six degrés de liberté. Les robots n'en comportent pourtant en général que quatre ou cinq. Mais il faut savoir que toute limitation des mouvements du poignet suppose plus de mouvements pour d'autres parties du bras.

La main du robot

Voyons maintenant la conception de la main. Idéalement, elle devrait ressembler à une main humaine terminant un bras humain. Certaines mains de robot s'approchent d'assez près de cette définition. Mais la forme la plus courante d'une main de robot est à trois « doigts » crochus destinés à agripper. Elle comporte alors deux « doigts » et un « pouce » en opposition. L'ensemble est capable de saisir des objets.

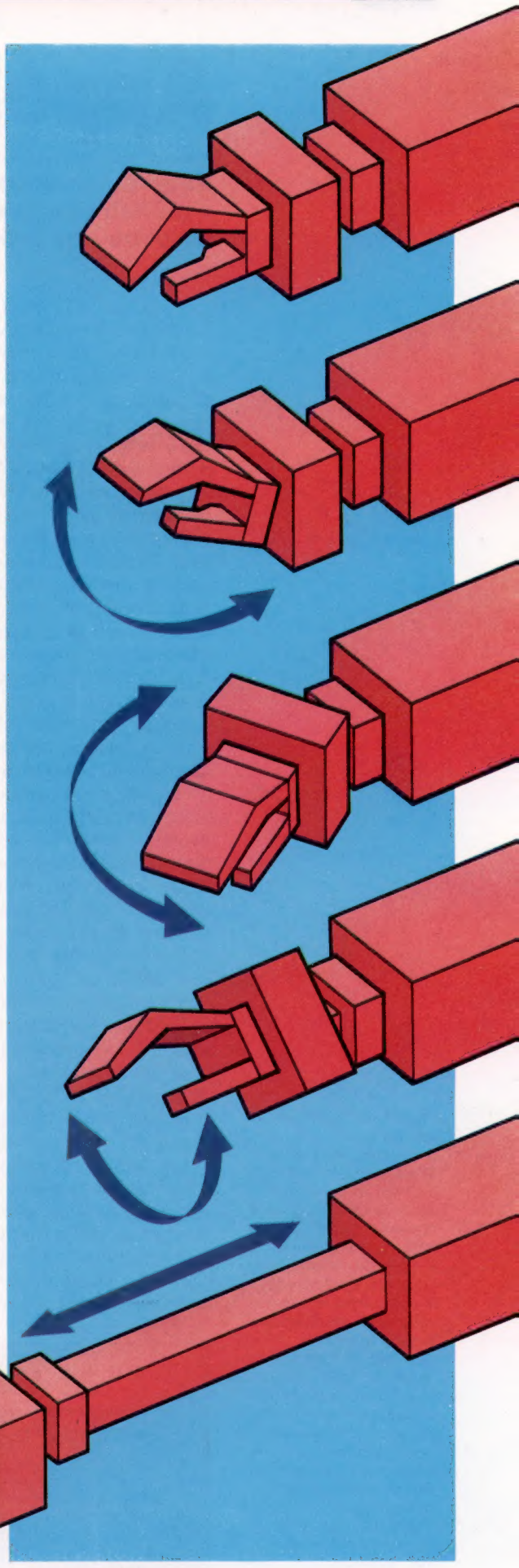
L'énergie qui anime cette main de robot peut être de l'un des trois types précédemment vus. Cela dépend des tâches qu'elle devra effectuer. S'il s'agit de déplacer de gros objets de plusieurs tonnes, l'énergie hydraulique sera appropriée. Mais pour beaucoup d'applications, l'énergie électrique ou pneumatique sera suffisante du fait que la main devra simplement agripper un objet et le relâcher le moment voulu. En outre, si le bras et le poignet ont positionné la main de manière appropriée, cette dernière pourra ne pas être très sophisitiquée : un simple mouvement d'ouverture et de fermeture à la manière d'une pince suffira.

Pourtant, dans de nombreux cas, le bras du robot ne comporte pas de main. Nous avons déjà parlé de l'extrémité du bras comme d'une main, mais elle peut ressembler à bien d'autres choses. Un robot utilisé pour effectuer des soudures n'a pas du tout besoin de main terminale — un fer à souder peut être ajusté directement au bras. En réalité, certains robots sont capables de choisir eux-mêmes leur extrémité selon la tâche à effectuer. Ils peuvent changer de terminaison (retirer le tournevis qu'ils avaient pour effectuer une certaine opération par exemple), et mettre en place une autre (un pistolet à peinture par exemple). L'adaptation de la terminaison se fait dans un socle au niveau du poignet. Cela ne ressemble pas vraiment au bras humain mais pour le robot, c'est un bon moyen de s'adapter.

Poignet de robot
Lorsque le « bras-deux-axes » est équipé d'un poignet souple, l'espace qu'il est susceptible de couvrir est beaucoup plus important. En revanche, les transformations de coordonnées sont beaucoup plus complexes. Mais l'aspect mathématique reste relativement simple, par rapport aux problèmes d'ingénierie, de légèreté, de résistance, de précision et de flexibilité. Le seul problème est de décider de la meilleure orientation de la main-pince pour prendre l'objet.

Pivot horizontal et vertical
Le poignet humain a les mêmes contraintes que celui du robot pour les déplacements horizontaux et verticaux : concilier la résistance et la miniaturisation avec la flexibilité et la légèreté. Ces articulations n'étendent pas le champ d'accès du bras, mais accroissent de manière sensible la maniabilté du bras pour saisir des objets concrets.

Enroulement et extension
Ces articulations sont beaucoup plus simples à construire et contribuent davantage à accroître le champ d'accès du bras que les pivots horizontaux et verticaux. Ces caractéristiques suffisent pour la majorité des applications industrielles.



Appel roulant

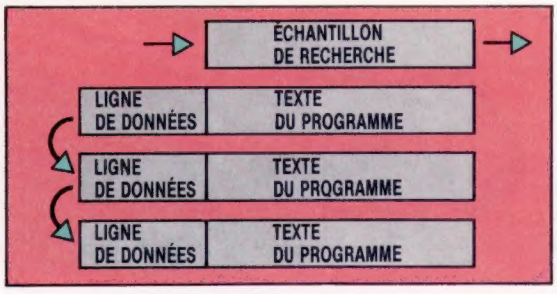
Nous commençons une série d'articles sur les programmes dits « utilitaires » des micros les plus répandus, et nous développerons en outre nos propres utilitaires BASIC.

La nature des utilitaires présents sur les micros domestiques varie énormément. Certains micros ne comportent qu'un seul utilitaire, l'éditeur. D'autres présentent une gamme d'utilitaires complète avec, par exemple, les commandes TRACE (tracer) et RENUMber (renuméroter). TRACE permet de suivre à la trace chaque ligne d'instruction BASIC en cours d'exécution en affichant son numéro de ligne. RENUM renumérote automatiquement les lignes d'un programme BASIC. Ces deux facilités sont extrêmement appréciables pour le développement d'un programme et la recherche des erreurs. Quels que soient les utilitaires fournis avec votre machine, il sera toujours important de pouvoir disposer d'autres facilités de programmation, et de nombreux programmes sont disponibles à cet effet sur le marché.

Les programmes utilitaires sont généralement écrits en langage Assembleur, à cause de la capacité du code machine à s'exécuter rapidement, et du fait qu'un programme en BASIC peut difficilement se modifier lui-même sans risquer de faire avorter l'exécution. Nous commencerons pourtant par quelques utilitaires simples que l'on peut écrire en BASIC. Nous pourrons ainsi nous consacrer à sa fonction, sans avoir à considérer d'autres détails annexes et compliqués, tels que le rôle du système d'exploitation et de l'interpréteur BASIC.

S'il est difficile pour un programme BASIC de se modifier lui-même, l'écriture d'un autre programme BASIC destiné à modifier le premier ne pose pas de problème. L'utilitaire que nous donnons ici cherche le nom d'une variable ou d'une fonction dans un programme BASIC, et affiche le numéro de ligne correspondant.

Les deux programmes cherchent d'abord où le texte du programme impliqué commence en mémoire. Ils le parcourent ensuite ligne par ligne, sautant les blocs qui ne peuvent manifestement pas comporter de nom, et ils extraient tous les noms. La phase finale consiste à comparer tous les noms recueillis avec celui demandé.



Recherche de fond en comble
Sur le BBC Micro et le Spectrum, une ligne de programme commence par 3 ou 4 octets pour le numéro et la longueur de la ligne. Vient ensuite le texte BASIC élémentaire (pour une ligne). Lorsque le programme rencontre la ligne, il enregistre son numéro et calcule l'adresse de départ de la prochaine ligne en se fondant sur la longueur de la ligne courante. Il passe alors l'échantillon de recherche dans le texte, jusqu'à rencontrer soit un message de fin de ligne ou de programme, soit une déclaration REM, soit le mot recherché.

Lorsque le programme entreprend d'examiner une nouvelle ligne de texte BASIC, il commence par noter le numéro de la ligne, qui est dans les deux cas stocké sur deux octets, et la longueur de la ligne (le nombre d'octets qu'elle occupe). Pour le programme du BBC Micro, la longueur de la ligne tient sur un octet et représente le nombre total d'octets entre le numéro de la ligne et le marqueur de fin de ligne (code ASCII 13). Pour le Spectrum, la longueur de la ligne est stockée sur deux octets et représente le nombre d'octets compris entre le caractère qui suit les octets destinés à la longueur de la ligne et le marqueur de fin de ligne.

Dans les deux versions, nous ignorerons toutes les instructions REM et tout ce qui figure entre guillemets, dans la mesure où il n'y a normalement pas de variables dans ces chaînes de caractères. Le BBC Micro vous permet de faire figurer des nombres hexadécimaux préfixés du caractère « & ». Il nous faut cependant faire en sorte que le programme ne les confonde pas avec des noms de variables. Il faudra donc qu'il saute toutes les chaînes précédées de «&». Par exemple, il faut éviter que notre programme ne prenne le nombre hexadécimal A0 dans &A0 pour le nom de variable «A0».

Pour le Spectrum, les nombres d'un programme sont stockés selon les équivalents en code ASCII de ses chiffres, suivis par l'octet du code ASCII 14, et de 5 octets contenant l'équivalent binaire du nombre. Notre programme doit pouvoir sauter le code du nombre et l'équivalent binaire sur 5 octets.

Ayant éliminé ces cas, le programme continue sa recherche de noms sur la ligne courante. Dans les deux programmes, un nom est défini comme commençant par une lettre et se poursuivant par d'autres lettres ou par des chiffres. La version du BBC Micro autorise des variables entières (identifiées par le signe «%» après le nom), ainsi que le caractère de soulignement. Les deux versions de notre programme autorisent les variables-chaînes suivies du caractère «$».

Le nom d'un tableau, d'une fonction ou (pour le BBC Micro) d'une procédure sera suivi d'un crochet ouvert ([). A proprement parler, cela ne fait pas partie du nom.

Le programme destiné au Spectrum est plus compliqué, car son BASIC ne fait pas la distinction entre majuscules et minuscules pour les lettres d'un nom de variable. Aussi FRED, Fred et FRed sont-ils considérés comme la même variable. Le programme du Spectrum convertit donc toutes

Spectrum

```
9000  INPUT "Nom à rechercher? "; LIGNE T$
9010  FOR I=1 TO LEN (T$)
9020  IF T$(I)>="a" AND T$(I)<="z" THEN
LET T$(I)=CHR$ (CODE (T$(I))-32)
9030  NEXT I
9040  LET ElémentpourREM=234
9050  LET Guillemet=34
9060  LET Nouvelleligne=13
9070  LET Souligné=95
9080  LET Nombre=14
9090  LET PROG=23635
9100  LET Pointeurdetexte=PEEK (PROG)+256*PEE
K (PROG+1)
9110  LET Ligneno=256*PEEK (Pointeurdetexte)+P
EEK (Pointeurdetexte+1)
9120  IF Ligneno>=9000 THEN STOP
9130  LET Pointeurdetexte=Pointeurdetexte+2
56*PEEK (Pointeurdetexte+1)
9150  LET Pointeurdetexte=Pointeurdetexte+2
9160  LET Prochaineligne=Pointeurdetexte+Longueurdetexte
9170  IF PEEK (Pointeurdetexte)=Nouvelleligne THEN
 LET  Pointeurdetexte=Pointeurdetexte+1: GO TO 91
10
9180  IF PEEK (Pointeurdetexte)<>ElémentpourREM
THEN   GO TO 9220
9190  REM Sauter la ligne REM
9200   LET Pointeurdetexte=Pointeurdetexte
9210  GO TO 9110
9220  IF PEEK (Pointeurdetexte)=Quote THEN
GO TO 9200
9230  REM Sauter tout ce qui est entre guillemets,
mais s'arrêter en fin de ligne s'il manque le
guillemet final.
9240  LET Pointeurdetexte=Pointeurdetexte+1
9250  IF PEEK (Pointeurdetexte)=Nouvelleligne THEN
 LET  Pointeurdetexte=Pointeurdetexte+1: GOTO 91
10
9260  IF PEEK (Pointeurdetexte)<>Guillemet THEN
GOTO 9240
9270  LET Pointeurdetexte=Pointeurdetexte+1
9275  GOTO 9170
9280  IF PEEK (Pointeurdetexte)<>Nombre THEN
 GOTO 9320
9290  REM Skip 5-byte Nombre binaire
9300  LET Pointeurdetexte=Pointeurdetexte+6
9305  GOTO 9170
9310  REM Le premier caractère du nom doit être une lettre
majuscule ou minuscule
9320  IF PEEK (Pointeurdetexte)>=CODE  ("A") A
ND PEEK (Pointeurdetexte)<=CODE  ("Z")  THEN
LET c$=CHR$ (PEEK (Pointeurdetexte)): GOTO
9370
9330  REM Utilisez des lettres majuscules
9340  IF PEEK (Pointeurdetexte)<=CODE ("a") A
ND PEEK (Pointeurdetexte)<=CODE ("z") THEN
LET c$=CHR$ (PEEK (Pointeurdetexte)-32): GO
TO 9370
9350  LET Pointeurdetexte=Pointeurdetexte+1
9360  GOTO 9170
9370  LET n$=""
9380  LET n$=n$+c$
9390  LET Pointeurdetexte=Pointeurdetexte+1
9400  REM Après le premier caractère du nom, une lettre,
un chiffre ou un souligné
9410  IF PEEK (Pointeurdetexte)>=CODE ("A") A
ND PEEK (Pointeurdetexte)<=CODE ("Z") THEN
LET c$=CHR$ (PEEK (Pointeurdetexte)): GOTO
9380
9420 REM Utilisez des majuscules au lieu de lettres
minuscules
9430  IF PEEK (Pointeurdetexte)>=CODE ("a") A
ND PEEK (Pointeurdetexte) =CODE ("z") THEN
LET c$=CHR$ (PEEK (Pointeurdetexte)-32): GO
TO 9380
9440  IF PEEK (Pointeurdetexte) =CODE ("0") A
ND PEEK (Pointeurdetexte) =CODE ("9") THEN
LET c$=CHR$ (PEEK (Pointeurdetexte)): GOTO
9380
9450  IF PEEK (Pointeurdetexte)=Souligné TH
EN  GOTO 9380
9460  REM Terminer les variables-chaînes par $
9470  IF PEEK (Pointeurdetexte)=CODE ("$") TH
EN LET n$=n$+"$": LET Pointeurdetexte=Pointeurdetexte
+1: GOTO 9500
9480  REM (Pour un tableau ou une fonction
9490  IF PEEK (Pointeurdetexte)=CODE ("(") TH
EN  LET n$=n$+CHR$ (PEEK (Pointeurdetexte)):
 LET Pointeurdetexte=Pointeurdetexte+1
9500  IF n$=t$ THEN PRINT n$; " A LA LIGNE "; Ligneno
9520  GOTO 9170
```

BBC Micro

```
30000  INPUT "Nom à rechercher" :
CIBLE$
30010  Elément pour REM=244
30020  Guillemet=34
30030  Hex=38
30040  Nouvelleligne=13
30050  Soulignement
30060  Pointeur de texte=PAGE
30070  Pointeurdetexte=Pointeurdetexte+1
30080  Ligneno=256*?Pointeurdetexte+?(Pointeurdetexte+1)
30090  IF Ligneno=32000 THEN END
30100  Pointeurdetexte=Pointeurdetexte+2
30110  Longueurdeligne=?Pointeurdetexte
30120  Findeligne=Pointeurdetexte+longueurdeligne-3
30130  Pointeurdetexte=Pointeurdetexte
30140  IF ?Pointeurdetexte=Nouvelleligne THEN GOTO
30070
30150  IF ?Pointeurdetexte<>ElémentpourREM THEN
GOTO 30180
30160    REM Sauter REM Ligne
30170    Pointeurdetexte=Derniéreligne:GOTO 30070
30180  IF ?Pointeurdetexte<>Guillemet THEN GOTO 3
0240
30190    REM Sauter tout ce qui est entre guillemets, mais
s'arrêter en fin de ligne lorsque le guillemet manque
30200    Pointeurdetexte=Pointeurdetexte+1
30210    IF ?Pointeurdetexte=Nouvelle ligne THEN GO
TO 30070
30220    IF ?Pointeurdetexte<>Guillemet THEN GOTO
30200
30230    Pointeurdetexte=Pointeurdetexte+1
30235  GOTO 30140
30240  IF ?Pointeurdetexte<>Hex THEN GOTO 303
00
30250    REM Sauter les nombres HEX, afin d'éviter toute
confusion avec les noms de variables
30260    Pointeurdetexte=Pointeurdetexte+1
30270    IF ?Pointeurdetexte=ASC("0") AND ?
Pointeurdetexte =ASC("9") THEN GOTO 30260
30280    IF ?Pointeurdetexte=ASC("A") AND ?
Pointeurdetexte  =ASC("F") THEN GOTO 30260
30285    GOTO 30140
30290  REM Le premier caractère d'un nom doit être une
lettre majuscule ou minuscule
30300  IF ?Pointeurdetexte>=ASC("A") AND ?
Pointeurdetexte<=ASC("Z") THEN GOTO 30330
30310  IF ?Pointeurdetexte>=ASC("a") AND ?
Pointeurdetexte<=ASC("z") THEN GOTO 30330
30320  GOTO 30130
30330  Nom$=""
30340  Nom$=Nom$+CHR$(?Pointeurdetexte)
30350  Pointeurdetexte=Pointeurdetexte+1
30360  REM Après le premier caractère, une lettre, un
chiffre ou un souligné
30370  IF ?Pointeurdetexte=ASC("A") AND ?
Pointeurdetexte<=ASC("Z") THEN GOTO 30340
30380  IF ?Pointeurdetexte  =ASC("a") AND ?Pointeurde
texte  =ASC("z") THEN GOTO 30340
30390  IF ?Pointeurdetexte>=ASC("0") AND ?Pointeurde
texte  =ASC("9") THEN GOTO 30340
30400  IF ?Pointeurdetexte=souligné THEN GO
TO 30340
30410  REM Terminer par $ pour une variable-chaîne, et
par % pour une variable entière
30420  IF ?Pointeurdetexte=ASC("0") THEN Nom
$=Nom$+CHR$(?Pointeurdetexte):Pointeurdetexte=
Pointeurdetexte+1:GOTO 30450
30430  IF ?Pointeurdetexte=ASC("%") THEN Nom
$=Nom$+CHR$(?Pointeurdetexte):Pointeurdetexte=
Pointeurdetexte+1
30440  REM ( Si tableau, procédure ou fonction)
30450  IF ?Pointeurdetexte=ASC("(") THEN Nom
$=Nom$+CHR$(?Pointeurdetexte):Pointeurdetexte=
Pointeurdetexte+1
30460  IF Nom$=CIBLE$ THEN PRINT Nom$;
"A LA LIGNE ";Ligneno
30470  GOTO 30140
30480  END
```

les lettres en majuscules avant de comparer les noms. Le Spectrum autorise des espaces dans les noms de variables, mais nous le déconseillons, car cela suscite souvent des problèmes.

Le BASIC du Spectrum ne fait pas strictement la distinction, courante pour la plupart des autres versions du langage BASIC, entre variables-chaînes et tableaux-chaînes. En réalité, une variable-chaîne sur cet appareil est plus semblable à un tableau de caractères. En fait, T$ et T$(i) se rapportent respectivement à une chaîne et à une partie de cette même chaîne. Le programme n'essaie pas de distinguer entre de simples variables-chaînes et des tableaux-chaînes. Il ne s'agit pas d'une limitation puisque le BASIC du Spectrum ne nous autorise pas à avoir une variable-chaîne et un tableau de même nom.

Pour utiliser notre programme utilitaire, saisissez-le et fusionnez-le au programme que vous voulez examiner, avec la commande MERGE pour le Spectrum. Appelez la routine de recherche par RUN 9000 (Spectrum), ou par GOTO 3000 (BBC Micro). Lorsque le programme vous le demande, donnez le nom que vous recherchez.

Les lutins de logo

Nous allons voir les principes de base pour créer des figures graphiques (lutins) en nous basant sur des exemples réalisés avec LOGO sur Commodore.

Avec LOGO, les lutins se manipulent comme la tortue et ils obéissent aux mêmes commandes. Cependant, à la différence de la tortue, nous pouvons en définir la forme nous-mêmes. Avec LOGO sur Commodore, la tortue est considérée comme le lutin numéro zéro ; sept autres figures restent donc disponibles. Par défaut, à l'initialisation, le lutin 0 (F0) est le « lutin courant » et suit les commandes saisies. Pour faire du lutin 1 (F1), par exemple, le lutin courant, il suffit de taper DIRE 1. Les commandes s'appliquent alors toutes au lutin jusqu'à ce qu'un autre soit retenu.

Une fois DIRE 1 tapé, vous ne verrez cependant rien apparaître à l'écran. La raison en est que toutes les figures, à l'exception de la tortue, sont d'abord cachées, (lève plume, LP). Pour voir apparaître F1, et la voir bouger, il faut taper CR (carré). Faites des essais avec cette grille en utilisant les commandes AV (avance), AR (arrière), DR (à droite), GA (à gauche), PP (pose plume), LP (lève plume), DT (démarre tortue), CT (cache tortue), etc.

Si vous déplacez le lutin 1 sur la position du lutin 0 (la tortue), vous vous apercevrez que F 1 est au-dessous de la tortue. De manière générale, les figures des numéros plus petits apparaissent au-dessus de celles du numéro plus grand. Cela se révèle utile pour des effets visuels à trois dimensions.

La disquette d'utilitaires LOGO du Commodore comporte un éditeur de lutins. Pour le charger, faites LIRE «EDLUT (lire « éditeur lutin). Pour éditer la forme du lutin 1, faites d'abord DIRE 1 pour qu'elle devienne le lutin courant, et tapez ensuite EDFOR (éditer forme). L'affichage représentera une vue agrandie de la grille à lutin et vous pourrez déplacer librement le curseur à l'écran. La touche astérisque remplira une case (1 pixel), et la barre d'espacement la laissera vide.

Votre figure dessinée, faites Contrôle-C pour définir sa forme. Si elle n'apparaît pas, tapez DT. La même forme peut être affectée à plusieurs figures. DONNEFORME 1 donnera à la figure courante la même définition que celle du lutin 1. Lorsque vous aurez défini un jeu de formes, vous pourrez sauvegarder les figures correspondantes sur un fichier par SAUVEGARDEFORMES «NOMDUFICHIER, et les charger à nouveau par CHARGEFORMES «NOMDUFICHIER.

Il existe un problème mathématique célèbre dans lequel quatre insectes (bugs/punaises) sont placés aux quatre coins d'un carré. Ils sont mis en mouvement selon la même vitesse, chacun suivant l'insecte à sa droite. L'objectif est de suivre leur cheminement. Nous donnons ici un programme LOGO qui met en œuvre ce problème avec des lutins.

Les procédures que nous donnons ici placent simplement dans chaque coin une copie du même lutin, et la mettent ensuite en mouvement, de sorte que chaque copie suive l'autre. La forme de l'insecte correspond au lutin 3, les autres figures recevant la même forme par DONNEFORME 3 dans la procédure de positionnement.

Notre solution est basée sur la procédure SUIVRE. Dans SUIVRE, X et Y sont d'abord affectées aux coordonnées x et y du lutin suivi (:B), puis le lutin suivant (:A) est dirigé sur ce point. Nous utilisons à cette fin la procédure VERS. Celle-ci prend en entrée deux données pour les coordonnées du point devant être dirigé vers la cible, et donne en résultat la direction à suivre.

Une utilisation intéressante des lutins est la création d'effets d'animation. On définit un ensemble de formes correspondant à un même objet. Chacune est légèrement différente, ce qui donne l'effet de mouvement lorsqu'elles sont exécutées ensemble. LOGO sur Commodore crée trois formes représentant grossièrement un homme en train de courir. Les procédures suivantes initialisent l'écran et positionnent ensuite les trois formes en mouvement.

```
POUR COURIR
    DIRE 0
    DESSINE
    CL
    GRANDX GRANDY
    DONNERDIRECTION90
    EXÉCUTION2
FIN

POUR EXÉCUTION :FORME
    AV 5
    DONNEFORME :FORME
    SI :FORME = 4 ALORS FAIRE «FORME 1
    EXÉCUTION :FORME + 1
FIN
```

Avant d'exécuter ces procédures, chargez le ficher LUTINS depuis le disque des programmes utilitaires. Ceux-ci comprennent des procédures très utiles, dont GRANDX et GRANDY, qui agrandissent de deux fois la taille d'un lutin, et PETITX et PETITY, qui la réduisent de moitié. Chargez les trois lutins en tapant CHARGEFORMES «COUREURS, et exécutez ensuite les procédures.

Nous définissons à la page suivante trois formes de lutin pour le prochain jeu.

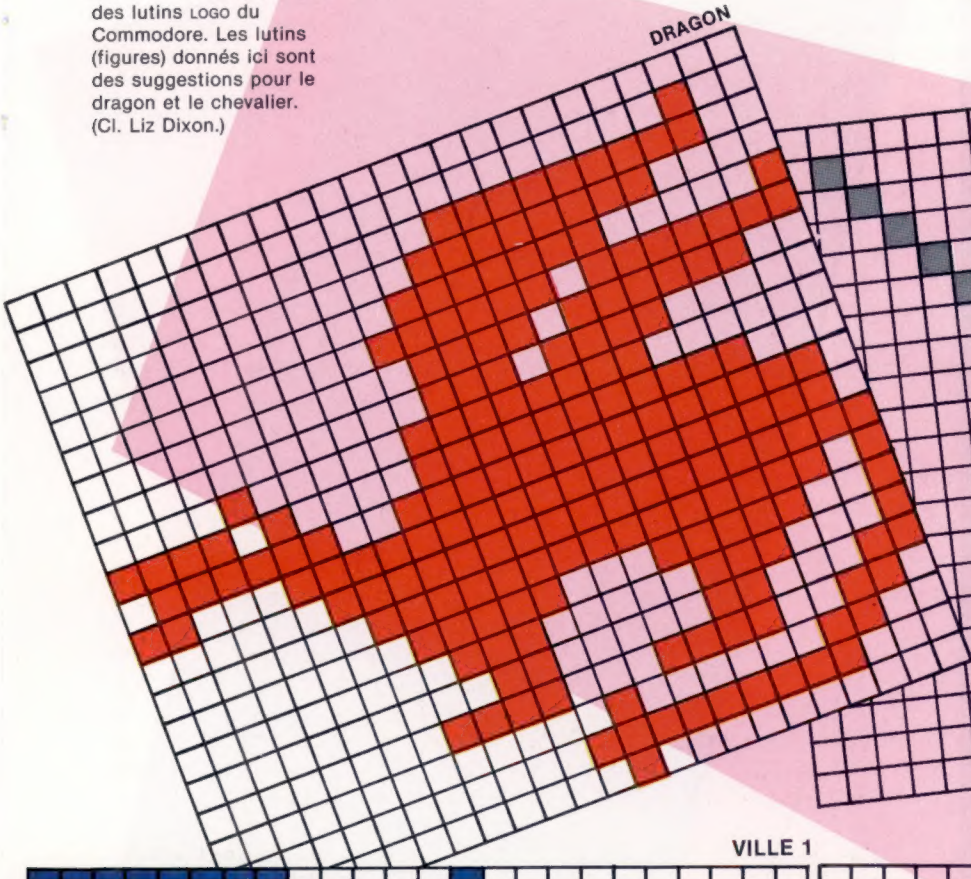
Variantes de logo

Ni le langage LOGO du Spectrum ni celui d'Apple n'utilisent de lutins.
Les utilisateurs d'Atari noteront bien ceci :
1. Il n'existe que quatre figures possibles.
2. Utiliser DONNEFR pour DONNEFORME.
3. L'éditeur de lutins est donné dans les instructions. La barre d'espacement appuyée remplit une position vide, et, inversement, une position pleine.

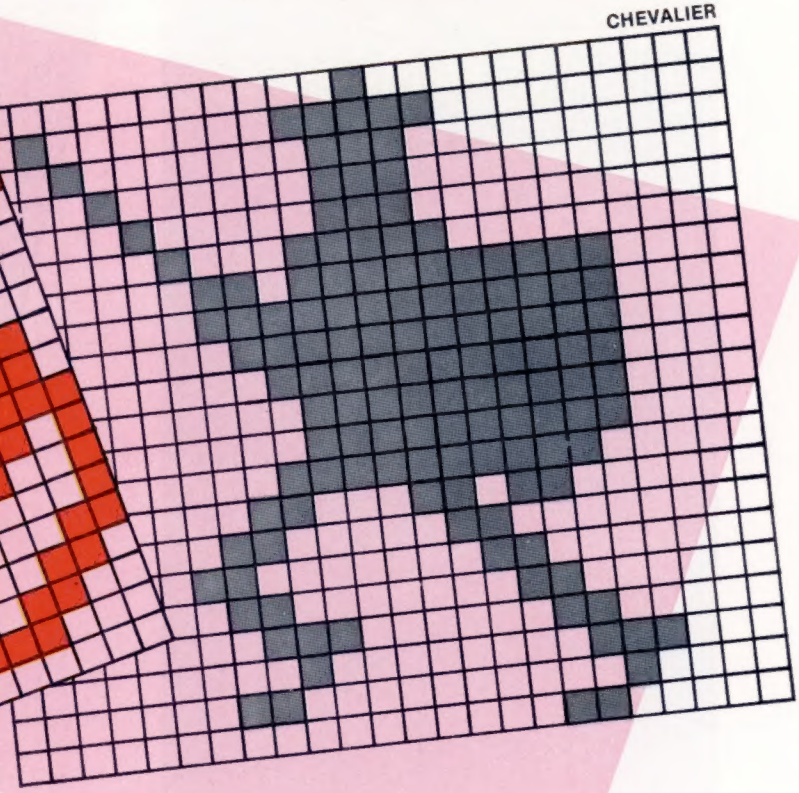
Dragon rampant

Nous présenterons ultérieurement le jeu du dragon et du chevalier, qui utilise les caractéristiques des lutins LOGO du Commodore. Les lutins (figures) donnés ici sont des suggestions pour le dragon et le chevalier. (Cl. Liz Dixon.)

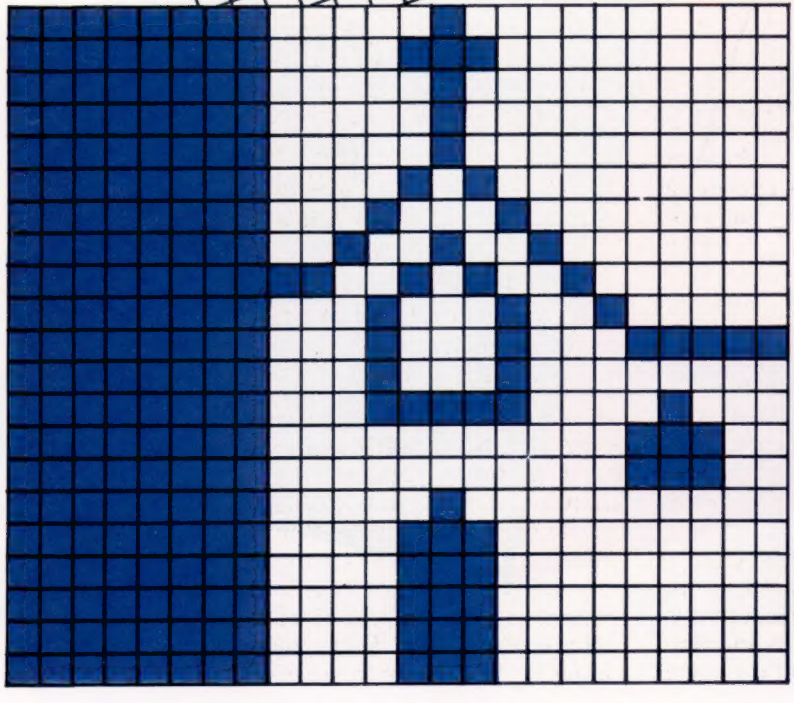
DRAGON



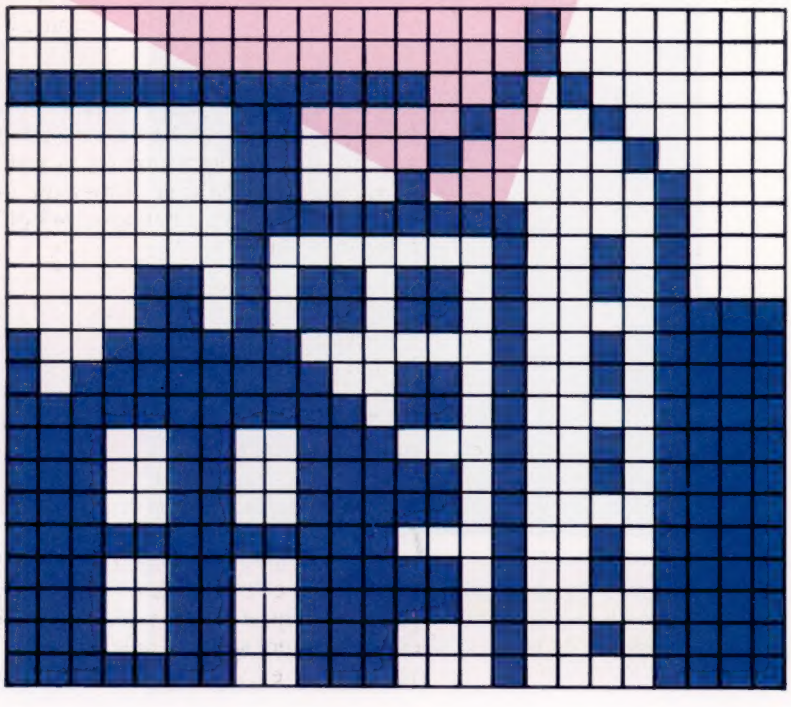
CHEVALIER



VILLE 1

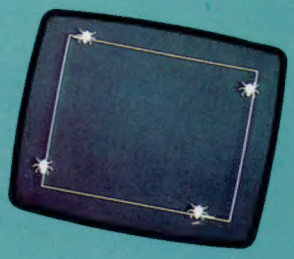


VILLE 2

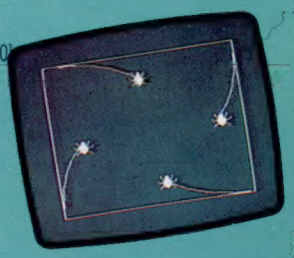


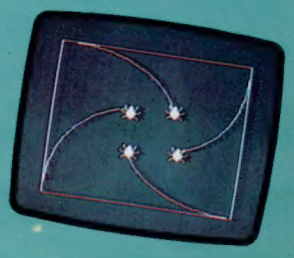
Quatre bugs

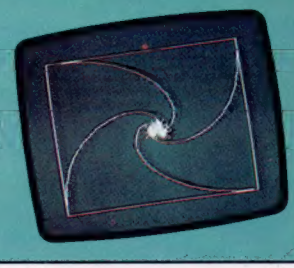
Dans cette démonstration géométrique classique, chaque insecte se dirige à tour de rôle directement sur la position de l'insecte à sa droite. Cet algorithme produit invariablement une spirale vers l'intérieur, dont chaque bras dessine la « courbe de poursuite », célèbre pour les pilotes de chasse et les amateurs de jeux d'arcades.



```
POUR BUGS
    INITIALISER
    DÉPLACER.BUGS
FIN

POUR INITIALISER
    DESSINER
    PLEIN.ÉCRAN
    DIRE 0
    CACHETORTUE
    LP
    DONNEXY (-100) (-100)
    CARRÉ 200
    POSITION 1 (-100) (-100)
    POSITION 2 (-100) 100
    POSITION 3 100 100
    POSITION 4 100 (-100)
FIN

POUR CARRÉ :COTÉ
    PP
    RÉPÈTE 4[AV :CÔTÉ DR 90]
    LP
FIN

POUR POSITION :0 :X :Y
    DIRE :0
    DONNEFORME 3
    LP
    DONNEXY :X :Y
    PP
    DT
FIN

POUR DÉPLACER.BUGS
    SUIVRE 1 2
    SUIVRE 2 3
    SUIVRE 3 4
    SUIVRE 4 1
    DÉPLACER.BUGS
FIN

POUR SUIVRE :A :B
    DIRE :B
    FAIRE «X XCOR
    FAIRE «Y YCOR
    DIRE:A
    DONNEDIRECTION :X :Y
    AV 10
FIN
```







Ian McKinnell

Exercice à trois bugs

Écrivez un programme LOGO pour un autre problème de bugs; cette fois, avec trois bugs aux sommets d'un triangle.

Projet d'alunissage

Voici le programme que nous proposons en solution au projet de programmation du dernier cours. Tapez ATTERRIR pour jouer à cette version simple du jeu :

```
POUR ATTERRIR
    INITIALISE JEU
FIN

POUR INITIALISER
    DESSINER.PLATEFORME RÉGLER.RÉTROFUSÉE
FIN

POUR TRACER.PLATEFORME
    CL DONNEXY (-20) (-60) CB DONNEXY 20 (-60) CL
FIN

POUR RÉGLER.RÉTROFUSÉE
    DONNEXY 0 120 FAIRE «VIT 0 FAIRE «KÉROSÈNE 50
FIN

POUR JOUER
    COMMANDE DÉPLACER
    SI YCOR <-53 ALORS BOOM STOP
    VIT.GRAVITÉ NIVEAU.KÉRO JOUER
FIN

POUR COMMANDE
    SI TOUCHE.LUE = «K ALORS METTRE.LES.GAZ
FIN

POUR TOUCHE.LUE
    SI RC? ALORS RÉSULTAT RC
    RÉSULTAT»
FIN

POUR METTRE.LES.GAZ
    SI :KÉRO > 0 ALORS FAIRE «VIT :VIT + 0,5 FAIRE «KÉRO
    :KÉRO -1
FIN

POUR DÉPLACER
    DONNEY YCOR + :VIT
FIN

POUR BOOM
    SI :VIT > (-1) ALORS AFFICHE [ATTERRISSAGE RÉUSSI,
    BRAVO] STOP
    AFFICHE [RATÉ, TOUT L'ÉQUIPAGE EST MORT!]
FIN

POUR GRAVITÉ
    FAIRE «VIT :VIT -0,2
FIN

POUR NIVEAU.KÉRO
    [AFFICHE «KÉRO :KÉRO]
FIN
```

Normalisation jaune

Plus d'une douzaine de sociétés japonaises ont conclu un accord pour réaliser un micro standard. Les premiers « MSX » arrivent sur le marché européen — le Sony Hit-Bit et le Toshiba HX-10.

La norme MSX impose, avec l'utilisation du microprocesseur Z80, le minimum de ROM (32 K) et de RAM (8 K), le type de puce graphique et sonore, le contenu du clavier (bien que la disposition puisse varier), le nombre minimum d'interfaces et leur conception, les écrans graphiques et texte, et, évidemment, le langage BASIC contenu en ROM. Puisque MSX représente une conception standard, les machines MSX devraient être toutes semblables. Les fabricants bénéficient toutefois d'une certaine souplesse au niveau de la qualité de mémoire supplémentaire, du type de clavier utilisé et du nombre d'interfaces supplémentaires. En pratique, Sony et Toshiba, comme la plupart des fabricants MSX, ont opté pour des caractéristiques situées au-delà des exigences minimales.

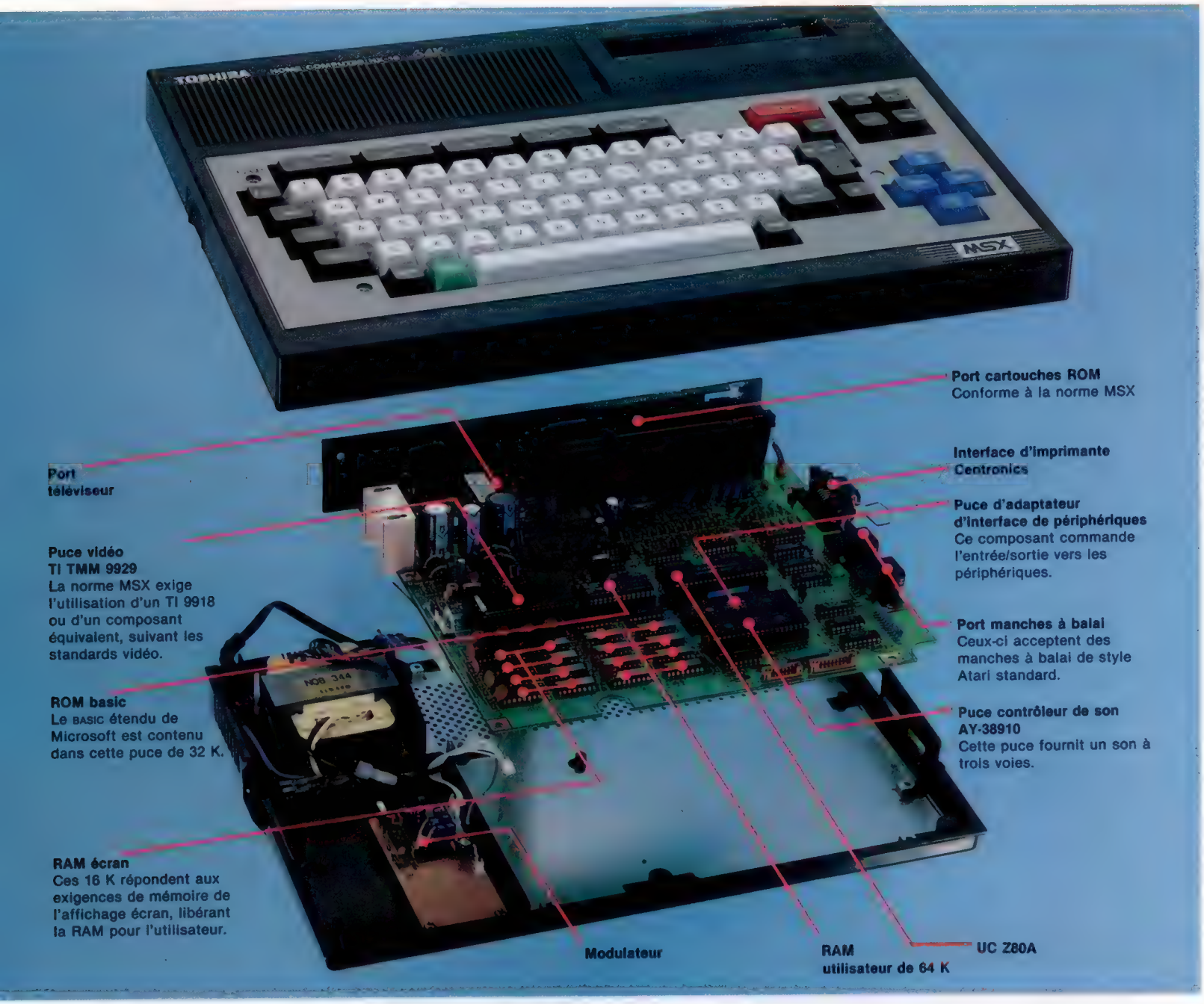
Le Sony Hit-Bit et le Toshiba HX-10 offrent tous deux d'excellents claviers, bien que certaines personnes trouvent les touches trop sensibles. Les deux micros sont livrés avec 64 K de mémoire principale; 16 K de RAM additionnelle sont dédiés à l'affichage vidéo. Les modèles Sony et Toshiba possèdent tous deux une interface imprimante Centronics et une paire de manches à balai, des éléments qui sont souvent vendus en option sur les ordinateurs domestiques.

Au départ, il était prévu que les ordinateurs MSX seraient vendus à très bas prix; mais les fluctuations monétaires et les coûts de fabrication ont fait grimper les prix. La hâte de commercialiser ces produits en Europe a aussi été responsable de cette augmentation de prix. Toshiba envoie toutes ses machines par avion plutôt que par bateau. La société a dû interrompre la production des versions japonaises de la machine pour se consacrer aux modèles européens en vue d'être le premier fabricant MSX à attaquer le marché.

L'une des premières choses que l'on remarque, lors de la mise sous tension d'un micro MSX, est la présence d'une ligne de mots affichée au bas de l'écran. Il s'agit de mots clés du langage BASIC, comme RUN, CLOAD», LIST, etc. Les micros ont 5 touches de fonction qui produisent ces mots fréquemment utilisés. Les mots affichés à l'écran servent d'étiquettes aux touches de fonction afin de permettre à l'utilisateur de se souvenir de la fonction de chaque touche.

Ces touches sont automatiquement définies lors de la mise sous tension des ordinateurs; mais il est facile de changer leur définition en utilisant la commande KEY. Bien qu'il n'y ait que cinq touches, dix fonctions peuvent être appelées en appuyant simultanément sur la touche SHIFT et sur la touche de fonction désirée. Lorsque SHIFT est appuyé, les indications affichées à l'écran changent pour fournir les nouvelles fonctions affectées aux touches. Chaque indication de fonction peut contenir jusqu'à quinze caractères, bien

Système standard
Le Toshiba HX-10 a deux ports manches à balai, une interface imprimante Centronics, un port cartouche ROM et un bloc touches de commande de curseur. Le BASIC MSX gère la commande par manche à balai de la même manière que la commande par curseur; les jeux peuvent donc être écrits pour l'un des deux types de commande.
(Cl. Chris Stevens.)



que ne soient affichés que les sept premiers de ceux-ci.

Le clavier et l'éditeur d'écran fonctionnent ensemble pour faciliter les corrections. Quatre touches permettent de déplacer le curseur; les changements peuvent alors être faits partout sur l'écran en écrasant simplement les caractères existants. La simple frappe d'une touche permet d'insérer et d'effacer des caractères. Les touches du curseur du Toshiba HX-10 ont les mêmes dimensions que les autres du clavier, mais le Sony Hit-Bit comporte des touches de curseur plus grosses que les autres. Puisque ces touches sont utilisées fréquemment, cette particularité peut se révéler très pratique.

Comme pour le matériel, le logiciel MSX comporte de nombreuses caractéristiques supplémentaires. Le BASIC MSX possède des commandes comme AUTO et RENUM et renferme plusieurs commandes servant à gérer le son, les graphiques et les interruptions. Il existe trois commandes primaires servant à créer des graphiques. LINE dessine une ligne entre deux points, et cette commande peut aussi servir à dessiner un carré en ajoutant la lettre B (pour « box ») après les coordonnées. Ajouter les lettres BF (pour « box fill ») dessine une surface carrée de couleur. La commande CIRCLE peut servir à dessiner des ellipses et des arcs en plus des cercles. Et la commande PAINT remplira d'une couleur toute zone délimitée, réussissant à traiter même les formes les plus irrégulières.

Le BASIC MSX comporte de nombreuses autres fonctions, bien que le jeu de commande le plus impressionnant — pour la gestion des interruptions — puisse ne pas être apprécié à sa

Avis à la population

Norme MSX

UC	Z80A, 3,58 MHz
RAM	Minimum 8 K
ROM	32 K incluant BASIC
ÉCRAN	16 couleurs, graphiques 256 × 192, 32 plans objets, affichage texte 40 × 24 ou 32 × 24 (puce vidéo TI 9918 ou équivalent)
SON	3 canaux, accessibles à partir de BASIC (puce contrôleur de son AY38910).
INTERFACES	Port cartouches MSX, sortie modulée de téléviseur, imprimante parallèle Centronics, interface cassette.
CLAVIER	Clavier QWERTY plus touches de fonction spéciales, 4 touches de curseur, 10 touches de fonction programmables.

Variantes MSX

SONY HIT-BIT	Logiciel de base de données intégré, sortie RGB, modules optionnels de RAM de 4 K.
TOSHIBA HX-10	Bus d'extension, 2 ports manche à balai.
YAMAHA CX-5	Mini-clavier musical et logiciel avec MIDI.
PIONEER	Interface de contrôleur de disque vidéo.
SANYO MPC100	Crayon optique et logiciel optionnels.
JVC HC7GB	Sortie RVB.
SPECTRAVIDEO SVI 728	Clavier numérique complet

Bien que la norme MSX exige un minimum de 8 K de mémoire, tous les fabricants ci-dessus mentionnés ont fourni une RAM utilisatrice de 64 K plus une RAM vidéo de 16 K dans leurs machines.

juste valeur au début. La gestion des interruptions est très importante en programmation graphique à haute vitesse. Il existe de nombreuses situations où un programme doit effectuer une tâche, tout en surveillant une autre action. Le programme Space Invader donne un bon exemple d'une telle situation. Il doit maintenir la mobilité des personnages sur l'écran, tout en enregistrant les mouvements du bouton de mise à feu. Il doit pouvoir faire deux choses à la fois, en passant rapidement d'une tâche à l'autre.

La solution MSX consiste à traiter certaines choses comme des événements. L'ordinateur reçoit l'instruction de rechercher un événement donné. Lorsque celui-ci survient, l'ordinateur passe automatiquement à un sous-programme qui gère cet événement.

L'écran graphique MSX peut afficher seize couleurs avec une résolution de 256 points par 192. Il est possible de définir jusqu'à trente-deux plans objets de 8 points par 8 (ou seize plans objets de 16 points par 16, ou huit plans objets de 32 points par 32). Pour bien exploiter les possibilités des plans objets, le BASIC MSX offre un jeu complet de commandes dédiées comme SPRITE, pour définir un plan objet, et PUT SPRITE, pour le positionner sur l'écran. Comme l'ont annoncé les fabricants MSX, un grand nombre de cassettes de programmes sont déjà offertes pour ces machines. Et la compatibilité promise se révèle réelle : le logiciel destiné au Toshiba HX-10 fonctionne parfaitement sur le Sony Hit-Bit, et vice versa. Cela est vrai tant pour les programmes sur cassettes que pour ceux sur cartouches.

Après des années d'incompatibilité, il semble presque magique de sortir une cartouche d'un ordinateur et de l'utiliser sur un autre. La société MSX compte sur cette situation pour constituer une vaste gamme de logiciels. Mais...

Interface standard

Chris Stevens

TOSHIBA HX-10 MSX

PRIX

★ ★ ★

DIMENSIONS

365 × 245 × 60 mm.

UC

Z80A, 3.58 MHz.

MÉMOIRE

RAM 64 K (28 K disponible pour BASIC), RAM vidéo 16 K, ROM 32 K incluant le BASIC.

ÉCRAN

40 colonnes × 4 lignes de texte, graphiques de 256 × 192, avec 16 couleurs et jusqu'à 32 plans objets.

INTERFACES

Imprimante Centronics, téléviseur, moniteur, sortie audio, 2 ports manche à balai, port cassette, connecteur de cartouche ROM, bus d'extension.

LANGAGE DISPONIBLE

BASIC Microsoft étendu.

CLAVIER

De type machine à écrire, à 68 touches avec bloc de touches de commande du curseur, plus 5 touches de fonction programmables.

DOCUMENTATION

Guide de montage et guide de référence de programmation. Ces deux manuels sont bien faits mais incomplets.

FORCES

Le BASIC MSX dispose de nombreuses fonctions utiles, dont d'excellentes commandes de son et de graphiques. La norme MSX est un élément positif parce qu'il implique un plus grand nombre de programmes et de périphériques disponibles.

FAIBLESSES

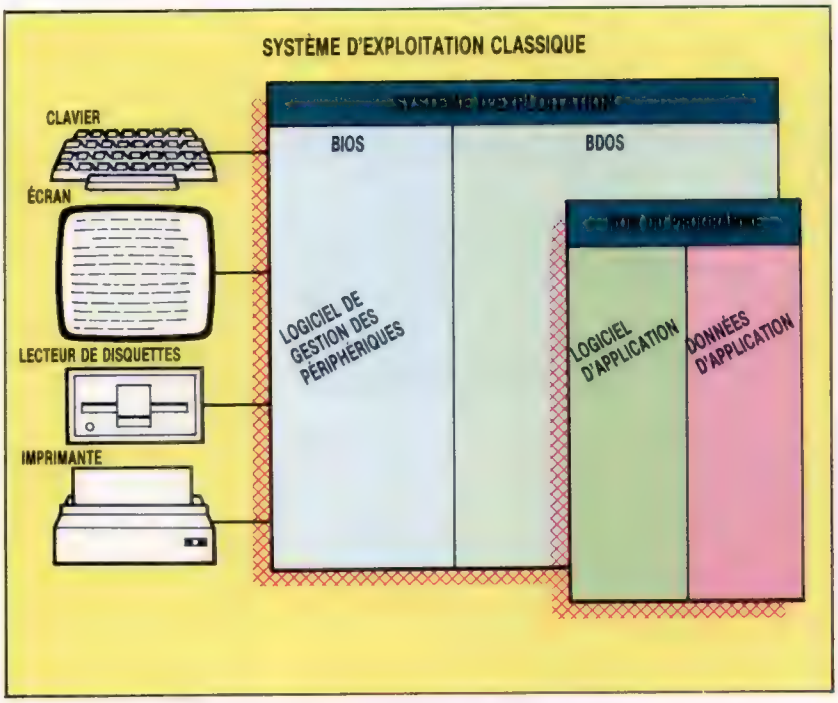
Le BASIC MSX n'offre aucune possibilité de programmation structurée; la norme et ses périphériques ne s'imposent que très lentement.

DIFFÉRENCES

Le Sony Hit-Bit offre 3 programmes intégrés en ROM, une sortie de moniteur RVB, et la disposition du clavier est légèrement différente.

Maître du monde

Plusieurs programmes réunis en un seul peuvent donner un logiciel intégré. Le gros défaut est de consommer énormément d'espace mémoire. Mais il existe une autre méthode.



Un ordre est un ordre
Dans un système d'exploitation de type classique, c'est le programme qui décide de tout : l'affichage écran, l'accès au lecteur de disquettes, l'interrogation du clavier. Le système d'exploitation proprement dit se borne à assurer l'exécution des directives du programme et veille au bon fonctionnement, extrêmement complexe et détaillé, du matériel. Ce n'est qu'un exécutant.

La conception d'un logiciel intégré peut partir du principe suivant : le système d'exploitation assure l'intégration proprement dite, et manipule des programmes séparés dont il gère automatiquement et simultanément la mise en route et le fonctionnement.

Il faut pour cela des matériels et des logiciels bien plus sophistiqués que ceux dont on fait usage ordinairement. La mise au point d'un tel système d'exploitation n'a pas été chose facile. En ce domaine, Apple a joué un rôle de pionnier avec le Lisa et le Macintosh. D'autres compagnies — Microsoft en particulier — s'efforcent actuellement de mettre au point des dispositifs analogues, capables de tourner sur des ordinateurs comme l'IBM PC.

La notion même de programme est à cette occasion profondément bouleversée. Une grande part du logiciel est consacrée à l' « interface utilisateur » : entrée des commandes et des informations fournies par l'usager, présentation des résultats. Il n'existe encore aucune norme reconnue, et les procédures d'exploitation varient avec chaque programme.

Un système d'exploitation intégré comporte un ensemble de routines destinées à gérer l'interface utilisateur pour chaque programme d'application. L'une d'elles, par exemple, a pour unique tâche d'afficher à l'écran toutes les options disponibles, afin que l'usager puisse faire son choix. Un des avantages du système est que les procédures d'exploitation seront en gros les mêmes quels que soient les programmes à manipuler. Il suffit donc de se familiariser avec l'un d'eux pour savoir comment mettre en œuvre les autres !

Une « souris » est un appareil à boule que l'on déplace sur la surface du bureau. On commande ainsi le positionnement du curseur sur l'écran, et on peut donc faire choix de telle ou telle option. L'écran tactile répond aux mêmes fonctions ; dans ce cas, une matrice constituée de rayons lumineux réagit au toucher d'un doigt. L'affichage se répartit entre plusieurs fenêtres, dont chacune est consacrée à une tâche particulière, ce qui exige un microprocesseur très rapide, énormément d'espace mémoire et une résolution graphique très élevée. Évidemment, les prix s'en ressentent. Mais cela en vaut la peine : le système est facile à maîtriser et à employer, il permet à l'usager de passer sans effort d'une option à l'autre, d'en faire tourner plusieurs en même temps, et surtout il peut accepter pratiquement tout programme déjà existant.

Le contrôle des opérations

De ce point de vue, il est intéressant d'étudier comment l'intégration des programmes est accomplie. L'utilisateur n'a jamais aucun contact direct avec eux ; c'est le système d'exploitation qui se charge de tout. Chaque application particulière n'est plus qu'une extension de l'ensemble du système d'exploitation, l'ordinateur ne représentant, pour sa part, qu'une sorte d'élément « intégré » unique.

Là est la seconde différence avec le procédé traditionnel, presque toujours à sens unique, qui consiste à charger le système d'exploitation de l'exécution d'une tâche spécifique définie par le programme.

Dans un système intégré, au contraire, les rôles sont inversés. C'est au programme de répondre aux exigences du système d'exploitation. Par exemple : redessiner l'affichage parce que l'utilisateur est passé de l'autre côté de l'écran ; s'interrompre parce qu'il faut passer à une autre application ; ou accueillir des données prélevées dans un tableur. En d'autres termes, le programme doit être capable de réagir à chaque interrogation ou demande du système d'exploitation, aussi bien que dans les autres méthodes.

Un tel degré de coopération rend facile la création d'un environnement intégré. Chaque pro-

gramme dispose de sa propre fenêtre à l'écran, et il suffit d'y placer le curseur, à l'aide de la souris, puis d'y choisir une option, pour qu'aussitôt le système d'exploitation en prenne note et lance les opérations qui s'imposent.

Par exemple, si vous vous placez dans un des coins d'une fenêtre et que vous choisissez l'option qui permette de la déplacer, des routines spécialisées effectueront le travail demandé, puis, si c'est nécessaire, informeront le programme des changements survenus, afin qu'il puisse modifier son affichage de façon appropriée.

Vous pouvez aussi passer à une autre fenêtre : dans ce cas, le système d'exploitation active un nouveau programme, après avoir mis en sommeil le premier — l'interruption entre les applications est aussi simple que le déplacement de la souris.

Tout comme les logiciels intégrés de type classique, de tels ensembles rendent à peu près inévitable les présences simultanées en mémoire de toutes les données et de toutes les applications. On aura donc besoin de beaucoup d'espace : un méga-octet pour Lisa, 512 K pour le Macintosh. Même ainsi, le système d'exploitation doit parfois procéder à des échanges d'informations avec l'extérieur — le lecteur de disquettes — si l'on veut qu'il puisse tout faire entrer en ligne de compte. Les exigences de rapidité imposent, en règle générale, le recours au disque dur.

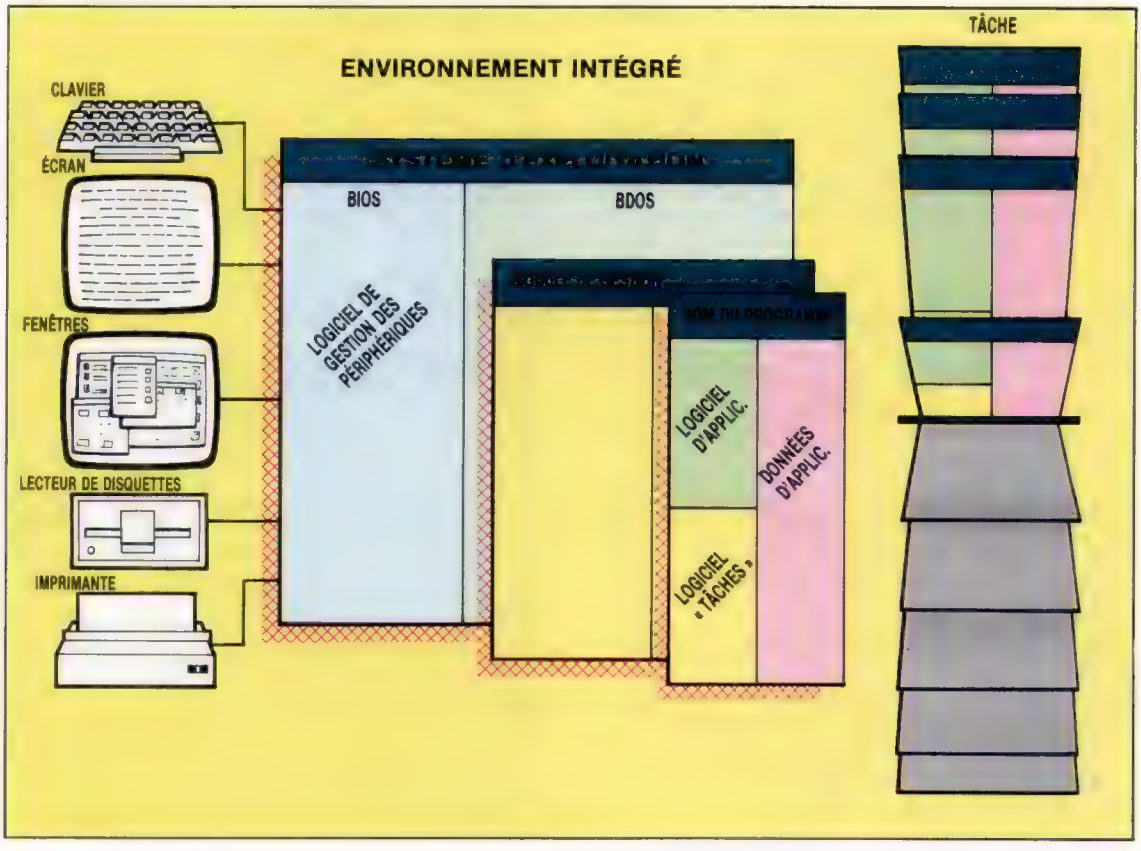
Pour procéder à des transferts de données, un ensemble de routines provoque d'abord l'interruption du premier programme (« exportateur ») et la mise en route du second (« importateur »). Celui-ci se verra alors enjoindre de lire les données en question et de les traiter. De telles voies de passage peuvent être établies de façon automatique. Par exemple, toute modification de l'information contenue dans un tableur entraîne aussitôt des changements dans le graphique correspondant. Bien entendu, les deux programmes ne tournent pas en même temps. C'est le système d'exploitation qui jongle avec eux en fonction de ses propres besoins.

Ouvrez les fenêtres

Le Lisa d'Apple fait usage d'une méthode encore plus raffinée : l'information affichée dans une fenêtre peut être « découpée » et « collée » sur une autre. Les instructions de formatage sont transmises en même temps que les données, et un graphique créé par un logiciel de gestion est très aisément transféré au sein d'un autre programme. Il suffisait d'y penser.

C'est là, sans doute, la manière la plus souple de créer un logiciel intégré. Vous pourrez mêler ou assortir tous les programmes qui vous seront nécessaires, passer de l'un à l'autre, y faire circuler vos éléments d'information, le tout avec la plus grande facilité. Un gros inconvénient, toutefois : il faut pour cela un matériel très évolué, qui reste encore très onéreux, et les logiciels se font attendre...

Une innovation technologique aussi décisive demande, il est vrai, un certain temps avant d'être communément acceptée. Les équipes de recherche de Xerox avaient mis au point la souris et les « fenêtres » il y a dix ans déjà. C'est maintenant seulement qu'elles sont enfin accessibles au grand public !



Des opérations combinées
Dans un ensemble intégré, le système d'exploitation se voit enrichi d'un module de gestion, pour qui les programmes et les données en mémoire sont autant de « tâches » différentes à accomplir. En fonction des exigences de l'utilisateur et des besoins liés à chacune d'elles, il les fait entrer et sortir de la mémoire principale ou du lecteur de disquettes. Un formatage prédéfini de l'information lui permet d'extraire et de transférer les données d'une application à l'autre, et donc de les répartir entre les « tâches ». Le système d'exploitation proprement dit, si détaillé qu'il soit, n'est plus qu'un simple support système pour le logiciel. (Cl. Kevin Jones.)

Réveille-matin

Nous avons déjà conçu et construit un boîtier relais. Nous allons étudier la conception de certains programmes qui utilisent ce boîtier relais et illustrent certaines applications domestiques simples.

Le boîtier relais de secteur est conçu pour contrôler l'alimentation du secteur en rapport avec tout dispositif qui lui est branché. En réponse à un signal basse tension, le boîtier transmet ou coupe le courant de la prise. Le mode de fonctionnement est tel que l'alimentation à la prise est maintenue tant qu'un courant de faible intensité est fourni au relais. Nous pouvons donc exciter le relais directement à partir de la sortie basse tension du boîtier que nous avons déjà construit. L'alimentation provenant du boîtier relais correspondra exactement au courant de faible tension fourni au relais par la sortie basse tension du boîtier. Par conséquent, la commande de l'alimentation secteur peut être réalisée en ayant recours aux mêmes techniques de programmation que celles utilisées pour commander les dispositifs à basse tension.

Si, par exemple, les conducteurs de faible tension du relais sont connectés aux bornes positives et négatives de la ligne 0 de la sortie du boîtier, et s'ils sont branchés dans une prise secteur, alors un courant secteur sera fourni à la prise du boîtier relais lorsque le bit 0 du registre de données du port utilisateur sera au niveau élevé. Lorsque le bit 0 est ramené au niveau bas, la prise du boîtier relais cesse de fournir l'alimentation secteur. Il est possible de brancher jusqu'à quatre boîtiers relais à la sortie du boîtier faible tension et de les piloter de cette façon.

Nous pouvons utiliser ce simple dispositif de commutation pour mettre au point plusieurs types de systèmes de commande. Essayons d'abord de programmer un ordinateur afin d'obtenir une réponse verbale en provenance d'un magnétophone lorsque nous appuyons sur un bouton.

Nous devons d'abord enregistrer une série de phrases, comme « vous appuyez sur mon bouton » suivi de « vous venez de le faire » et « je vous avertis », et ainsi de suite. Après avoir enregistré les messages, vous devez connecter le bouton et le magnétophone à votre port utilisateur et écrire un programme qui sollicite les phrases l'une après l'autre en réponse à une pression sur le bouton.

Nous devons faire les connexions suivantes vers le port utilisateur :

1. Brancher les conducteurs du relais secteur aux bornes positive et négative de la ligne 0 sur la sortie du boîtier basse tension.
2. Brancher le cordon d'alimentation du boîtier relais de secteur dans une prise murale et mettre celle-ci sous tension.
3. Connecter les deux conducteurs du bouton aux bornes positive et négative de la ligne 7 du boîtier tampon.

La principale mesure que nous devons prendre lors de la conception du logiciel d'exploitation de ce sytème consiste à nous assurer que le magnétophone sera commandé avec précision lorsqu'un message sera produit. Avant d'écrire le programme, nous devons donc chronométrer avec précision chaque message, et entrer ces données dans le programme de commande. Le chronométrage peut être réalisé à l'aide de l'horloge interne du micro ou à l'aide d'un chronomètre. La bande comporte trois phrases d'une durée de T(1), T(2) et T(3) secondes; nous pouvons donc écrire un programme qui, après que nous avons appuyé sur le bouton, met le magnétophone en fonction pendant la période exacte correspondant à la durée de chaque message. Si le chronométrage des phrases est fait avec précision, chaque phrase devrait com-

BBC Micro

```
 10 REM PROGRAMME ENREGISTREMENT BBC
 20 DIM T(3)
 30 DDR=&FE62:DATREG=&FE60
 40 ?DDR=127:REM L7 ENTREE
 50 ?DATREG=0:REM ARRET TOTAL
 60 CLS
 70 FORI=1TO3
 80 INPUT"DUREE (SECS)":T(I)
 90 NEXT I
100 :
110 FORL=1TO3
120 CLS
130 REPEAT
140 UNTIL(?DATREG AND 128)=0:REM L7
150 ?DATREG=1:REM MAGNETO EN MARCHE
160 TIME=0:REM DEBUT
170 REPEAT
180 UNTIL TIME>T(L)*100
190 ?DATREG=0:REM ARRET MAGNETO
200 NEXT L
210 END
```

Commodore 64

```
10 REM PROGRAMME ENREGISTREMENT CBM64
20 DD=56579:DATREG=56577
30 POKEDDR,127:REM ENTREE L7
40 POKEDATREG,0:REM ARRET TOTAL
50 PRINTCHR$(147): REM EFFACER ECRAN
60 FORI=1 TO 3
70 INPUT"DUREE (SECS)";T(I)
80 NEXT I
90 :
100 FOR L=1TO3
110 IF(PEEK(DATREG)AND128)<>0 THEN 110
115 POKETRAG,1:REM MAGNETO EN MARCHE
120 T=TI:REM INITIALISER MINUTERIE
130 IF T(L)>(TI-T)/60 THEN130
140 POKEDATREG,0:REM ARRET MAGNETO
150 NEXT L
160 END
```

mencer à son début lors des mises en fonction successives du magnétophone.

Les programmes suivants (pour le Commodore 64 et pour le BBC Micro) solliciteront le fonctionnement du magnétophone pendant trois périodes successives — T(1), T(2) et T(3) — en réponse à la pression du bouton. Vous devez trouver la valeur de ces variables correspondant à vos enregistrements.

Après avoir mis au point ce système, examinons un projet qui vous permettra de transformer votre micro en un réveille-matin programmable. Un tel système doit bien sûr être adapté aux besoins de chacun. Le programme que nous donnons permet à l'utilisateur de spécifier :

1. L'heure.
2. Le nombre d'intervalles de « sieste » (périodes entre les alarmes ou la musique) requis.
3. Pour chaque intervalle de sieste, si cette période doit être constituée de musique, d'alarme ou d'un moment de silence, et la longueur de l'intervalle.
4. Si une lumière doit être allumée pour chaque intervalle.
5. La dernière limite pour le réveil.

Ce programme fonctionne en supposant que les connexions suivantes ont été faites aux sorties du boîtier basse tension :

1. Un magnétophone est connecté à la ligne 0 en passant par un relais secteur.
2. Une lampe de chevet est à la ligne 1 en passant par un relais secteur.
3. Une sonnerie électrique de 9 V est connectée directement à la ligne 3.

Le programme accepte la dernière limite pour le lever et fait un compte à rebours des intervalles programmés afin de calculer le moment de départ pour chaque intervalle. Des tableaux sont utilisés pour stocker les données qui indiquent quels appareils sont mis en fonction pendant une période déterminée. Notez que les variables de tableaux reçoivent des valeurs qui correspondent à la valeur du bit requise dans le registre de données pour mettre en fonction tel appareil particulier. En utilisant l'instruction logique OR, nous pouvons trouver facilement le total qui doit être placé dans le registre de données pour mettre en fonction toute combinaison d'appareils.

Nous avons maintenant développé un système d'entrée et de sortie véritablement souple qui permet de commander, par micro-ordinateur, des diodes électroluminescentes, des dispositifs à basse tension et des appareils branchés au secteur, et qui entraîne également le micro à accepter et à interpréter des données d'entrée provenant de divers capteurs. De nombreuses possibilités s'offrent maintenant à nous pour concevoir des systèmes de commande correspondant exactement à nos besoins. Dans les exemples que nous donnons, le micro sert de minuterie programmable perfectionnée. D'autres applications pourraient aussi impliquer la mise sous tension et la mise hors tension de plaques électriques en réponse aux informations fournies par deux capteurs thermiques.

Ian McKinnell

Commodore 64

```
100 REM *** REVEILLE-MATIN CBM 64 ***
110 DDR=56579:DATREG=56577
120 POKE DDR,255:POKEDATREG,0
130 PRINTCHR$(147):REM EFFACER L'ECRAN
140 INPUT"NOMBRE D'INTERVALLES DE SIESTE";N
150 M=N+1
160 DIM A(M),M(M),L(M),TS$(M),T(M)
170 :
180 REM *** DONNEES INTERVALLE ***
190 FOR C=1 TO N
200 PRINT:PRINT"INTERVALLE NUMERO";C
210 INPUT"MUSIQUE, ALARME OU SILENCE (M/A/S)";AN$
215 AN$=LEFT$(AN$,1)
220 IF AN$<>"M"ANDAN$<>"A"ANDAN$<>"S" THEN 210
230 IF AN$="M" THEN M(C)=1:A(C)=0
240 IF AN$="A" THEN A(C)=0:M(C)=0
250 IF AN$="S" THEN A(C)=0:M(C)=0
260 INPUT"LUMIERE ALLUMEE (O/N)";AL$
270 L$=LEFT$(AL$,1)
280 IF L$<>"O" AND L$<>"N" THEN 260
290 IF L$="O" THEN L(C)=2:GOTO310
300 L(C)=0
310 INPUT"TEMPS INTERVALLE (MINS)";T(C)
320 NEXT C
330 :
340 INPUT"DERNIERE LIMITE POUR LE LEVER (HHMM)";LT$
350 LT$=LT$+"00":REM AJOUTER SECONDES
360 TS$(N+1)=LT$:REM DERNIERE LIMITE
370 REM CONVERTIR LA DERNIERE LIMITE EN MINUTES
380 LM=60*VAL(LEFT$(LT$,2))+VAL(MID$(LT$,3,2)
390 :
400 INPUT"HEURE ACTUELLE (HHMM)";TN$
410 TI$=TN$+"00":REM DEMARRER MINUTERIE
420 :
430 REM *** ANALYSE ET CALCUL ***
440 REM *** CALC TEMPS DE DEPART INTERVALLES ***
450 FOR C=N TO 1 STEP -1
460 LM=LM-T(C):REM DEMARRER TEMPS EN MINUTES
470 HR=INT(LM/60)
480 MN=INT(60*(LM/60-HR+.000001))
490 HR$=STR$(HR):REM HEURES
500 MN$=STR$(MN):REM MINS
510 MN$=MID$(MN$,2,LEN(MN$))
520 HR$=MID$(HR$,2,LEN(HR$))
530 REM ** AJOUTER ZEROS EN TETE **
540 SP$="00"
550 HR$=LEFT$(SP$,2-LEN(HR$))+HR$
560 MN$=LEFT$(SP$,2-LEN(MN$))+MN$
570 TS$(C)=HR$+MN$+"00"
580 NEXT C
590 :
600 REM **** GO ****
610 PRINTCHR$(147)
620 FOR C=1 TO N+1
630 IF TI$(TS$(C)THENGOSUB710:GOTO630
640 DN=M(C) OR A(C) OR L(C):REM DATREG DATA
650 [illegible]
670 NEXT C
680 [illegible]
700 :
710 REM ** S/P AFFICHAGE MINUTERIE **
720 PRINTCHR$(145);:REM CRSR HAUT
730 PRINTLEFT$(TI$,2);":";MID$(TI$,3,2);
740 PRINT" "+RIGHT$(TI$,2)
750 RETURN
```

BBC Micro

```
10 REM REVEILLE-MATIN BBC
15 MODE7
20 DR=&FE62:DATREG=&FE60
30 CLS
40 INPUT"NOMBRE D'INTERVALLES DE SIESTE";
45 M=N+1
50 DIM A(M),M(M), L(M),T(M),TS(M)
70 REM *** DONNEES INTERVALLE ***
80 FORC=1 TO N
90 PRINT"NUMERO INTERVALLE";C
95 REPEAT
100 PRINT"MUSIQUE, ALARME OU SILENCE";
105 INPUT"(M/A/S)";AN$
105 AN$=LEFT$(AN$,1)
110 UNTIL AN$="M"ORAN$="A"ORAN$="S"
120    IF AN$="M"THEN M(C)=1:A(C)=0
130    IF AN$="A"THEN M(C)=0:A(C)=0
140    IF AN$="S"THEN M(C)=0:A(C)=0
150    [illegible]
160      INPUT"LUMIERE ALLUMEE (O/N)";AL$
170      AL$=LEFT$(AL$,1)
180      UNTIL AL$="O" OR AL$="N"
190    IF AL$="O" THEN L(C)=2 ELSE L(C)=0
200    INPUT"TEMPS INTERVALLE (MINS)";T(C)
210    NEXT C
220 :
230 INPUT"DERNIERE LIMITE POUR LE LEVER (HHMM)";LT$
232  TS(N+1)=6000+(60*VAL(LEFT$(LT$,2)))
240  TS(N+1)=6000+(60*VAL(LEFT$(LT$,2)))
234   TS(N+1)=TS(N+1)+VAL(RIGHT$(LT$,2))
236 REM CONVERTIR DERNIERE LIMITE EN MINUTES
237 LM=60*VAL(LEFT$(LT$,2))
239  LM=LM+VAL(RIGHT$(LT$,2))
240 INPUT"HEURE ACTUELLE (HHMM)";TN$
250  TIME=6000+(60*VAL(LEFT$(TN$,2)))
255   TIME=TIME+VAL(RIGHT$(TN$,2))
260 :
270 [illegible]
280 FORC=N TO 1 STEP -1
290   LM=LM-T(C):REM DEBUT INTERVALLE
300   TS(C)=6000+LM
310   NEXT C
320 :
330  REM **** GO ****
340 CLS
350 FOR C=1 TO N+1
360   REPEAT
370     PROCnor(:ooo
380     UNTIL TEMPS =TS(C)
390   REGDATA=M(C) OR A(C) OR L(C)
400   ?DATREG=REGDATA
420   NEXT C
430 ?DATREG=0
440 END
999 :
1000 DEF PROCnor(:ooo
1020 MIN=(TEMPS DIV 6000) MOD 60
1030  HR=(TEMPS DIV 6000) MOD 60
1040 MIN$=STR$(MIN):HR$=STR$(HR)
1042 REM AJOUTA ZEROS EN TETE
1043 [illegible]
1044 HR$=LEFT$(SP$,2-LEN(HR$))+HR$
1045 MIN$=LEFT$(SP$,2-LEN(MIN$))+MIN$
1050 PRINTTAB(18,12)HR$":"MIN$
1060 [illegible]
```

Du micro à la compo

Il existe des logiciels qui permettent d'obtenir en sortie d'imprimante, non seulement un texte parfait, mais aussi une mise en page soignée, incluant figures ou tableaux, prête pour la photocomposition.

Depardon-Uzan/Gamma

A. Borland/Gamma

La composition d'un journal, ou même d'une plaquette publicitaire, nécessite un grand nombre d'étapes successives. Tout d'abord, la saisie du texte est effectuée par l'auteur — journaliste ou autre — sur son lieu de travail ou à son domicile, à partir de sa machine à écrire ou bien, dans le meilleur des cas, de son micro-ordinateur personnel muni d'un logiciel de traitement de texte. Ce texte doit ensuite être calibré, on choisit la police de caractères, des corps différents pour les titres et intertitres. Le tout est envoyé à l'imprimeur qui, après une seconde saisie, en réalise une épreuve appelée « placard », à partir de laquelle sera effectuée la maquette de chaque page, incorporant les documents annexes : schémas, tableaux, clichés... Celle-ci est renvoyée à l'imprimeur, qui réalise enfin le document définitif pour la photocomposition.

Depuis peu, il existe des systèmes comprenant un ou plusieurs ordinateurs ou micro-ordinateurs, munis d'un logiciel de traitement de texte et une photocomposeuse, qui réduisent toutes ces étapes à une seule, diminuant par là même les coûts et surtout la durée de la composition.

Mais la mise au point de la liaison entre les différents éléments entrant en jeu dans un tel système demande du temps et surtout une sérieuse étude préalable.

Le système peut comprendre plusieurs consoles. Ainsi, la saisie et la correction des textes se font sur les lieux mêmes de la création : chaque

De la dactylo... à la photocompo

Le « CompoType » est un clavier de pré-photocomposition qui permet à une simple dactylo de « frapper » du texte qui doit être photocomposé.
« CompoText », éditeur de texte, qui s'apprend en un quart d'heure, permet d'afficher votre texte à l'écran vidéo et de le rédiger ou le corriger à votre convenance.

Les symboles revenant couramment en photocomposition, tels INSERT SPACE ou THIN SPACE, sont mémorisés dans le clavier pré-programmé en une seule touche par symbole.
« CompoPrint », programme d'impression d'un listage pour auto-contrôle, permet non seulement de visualiser votre texte sur papier mais également de faire apparaître graphiquement les symboles spéciaux, de même que les grandes familles de gras, d'italiques, d'« extended », etc.

Un lecteur de disquettes 5 1/4 pouces, identique à ceux répandus couramment dans le monde de la micro-informatique, vous permet de mémoriser, corriger et copier vos textes. Sur demande, des lecteurs supplémentaires, voire des disques durs pour gros stockages, peuvent être adjoints au système.

Un réseau de « conseiller-dealers », équipé de CompoTypes couplés aux plus puissants systèmes de photocomposition du marché, est à même de lire vos disquettes et d'en extraire automatiquement vos textes photocomposés. Vous réalisez ainsi une double économie quant au prix et au délai de livraison.

Un peu de vocabulaire

• *La saisie au kilomètre*
Le texte est saisi sur une machine de traitement de texte, sans qu'on y apporte aucune indication complémentaire, quant à sa forme future. Il s'agit d'un enregistrement de tous les codes ASCII.
Pour l'opérateur qui saisira ce texte, il n'y aura aucune tâche supplémentaire : pas de centrage, ni de renfoncement à l'aide du tabulateur ou de la barre d'espace, etc. Ce texte sera transmis tel quel au « façonnier », qui lui fera subir divers traitements.

• *La justification*
Un texte dactylographié occupe plus de place qu'un texte composé, et les caractères ont une « chasse » fixe. Le fait d'aligner le texte à gauche ou/et à droite s'appelle la justification. Cette opération nécessite une chasse variable et l'incorporation dans une ligne d'espaces supplémentaires.
Il est également nécessaire de respecter un certain nombre de règles typographiques réunies dans le « code typographique » : les coupures en fin de ligne ne peuvent s'effectuer en n'importe quel endroit du mot, ni au milieu d'un nombre à plusieurs chiffres, etc. Un clavier classique de machine de traitement de texte ne dispose généralement pas de toutes les fonctions correspondant à ces règles. Ainsi, un texte saisi de façon décentralisée puis traité et restitué par une photocomposeuse risque de comporter des coupures non souhaitées ou bien des alignements étranges. De même, des colonnes de chiffres saisis avec des tabulations comporteront souvent des défauts d'alignement.

• *Le traitement typographique*
Ce traitement consiste à faire calculer, en fonction des différentes données du texte saisi, les nouvelles coupures de lignes, les coupures de mots éventuelles, etc. Cette tâche est confiée au façonnier.

• *L'enrichissement*
On distingue deux formes d'enrichissement. Le texte peut être enrichit soit par le façonnier, à l'aide d'une copie préparée, soit directement par le client, à l'aide d'un logiciel spécialisé. Dans le premier cas, c'est le façonnier qui ajoute les indications nécessaires de graisse, de corps, de justification, d'interlignage, etc. Cette situation est proche de la photocomposition classique, à ceci près que la saisie proprement dite est effectuée par le client. Dans le cas de l'enrichissement complet, le façonnier n'aura plus qu'à effectuer l'opération de restitution, puisque toutes les indications de corps, caractères, etc. sont écrites par le client ou incorporées dans le logiciel, au moment de la saisie.

• *Les balises*
Les balises sont des caractères introduits au moment de la saisie, pour indiquer une présentation, un « accident typographique », etc. Elle peuvent être constituées par la combinaison de quelques touches qui prendront leur sens ultérieurement. Par exemple, #$1 peut être enregistré avant chaque titre, #$2 avant chaque sous-titre, #$3 avant chaque note en bas de page, #$0 avant le texte proprement dit. Le façonnier devra, dans ce cas, transformer ces balises en chaînes d'instructions qui permettront le traitement et la restitution par la photocomposeuse. Ce procédé présente l'avantage que le texte ainsi enregistré par le client peut, jusqu'au dernier moment, changer d'apparence, sans qu'il soit nécessaire de modifier quoi que ce soit sur l'enregistrement initial. De plus, un même texte source peut donner naissance à différents produits imprimés, suivant la signification des balises.

• *La photocomposeuse*
C'est un périphérique sur lequel le façonnier envoie le texte une fois préparé à l'aide des différents traitements évoqués précédemment. On obtiendra alors soit un « bromure », soit un film, qui, selon le degré d'élaboration du logiciel utilisé, servira au montage manuel ou bien correspondra à une page entièrement montée.

auteur tape son texte sur son micro-ordinateur personnel ou sur un terminal. Les données sont alors stockées sur disquettes ou bien envoyées par liaison téléphonique vers l'ordinateur central et la photocomposeuse.

Les disquettes le plus souvent acceptées sont des 8 pouces au format IBM 3740. Le problème essentiel est la compatibilité entre les appareils. Certaines sociétés de photocomposition acceptent des disquettes IBM, Apple, Xerox, Tandy et Wang, d'autres peuvent recevoir à partir de micro-ordinateurs Commodore, Sirius ou IBM PC. Les firmes qui ont adopté le matériel Wang disposent en outre d'une photocomposeuse de la même marque.

Lorsque la solution adoptée est la liaison téléphonique, le problème de l'incompatibilité n'est pas pour autant résolu : il faut toujours établir une table de concordance entre le matériel de saisie utilisé et la photocomposeuse; mais ce type de liaison a au moins l'avantage d'éliminer le problème des incompatibilités physiques entre disquettes : format 8 pouces ou 5 1/4 pouces, sectorisation, etc.

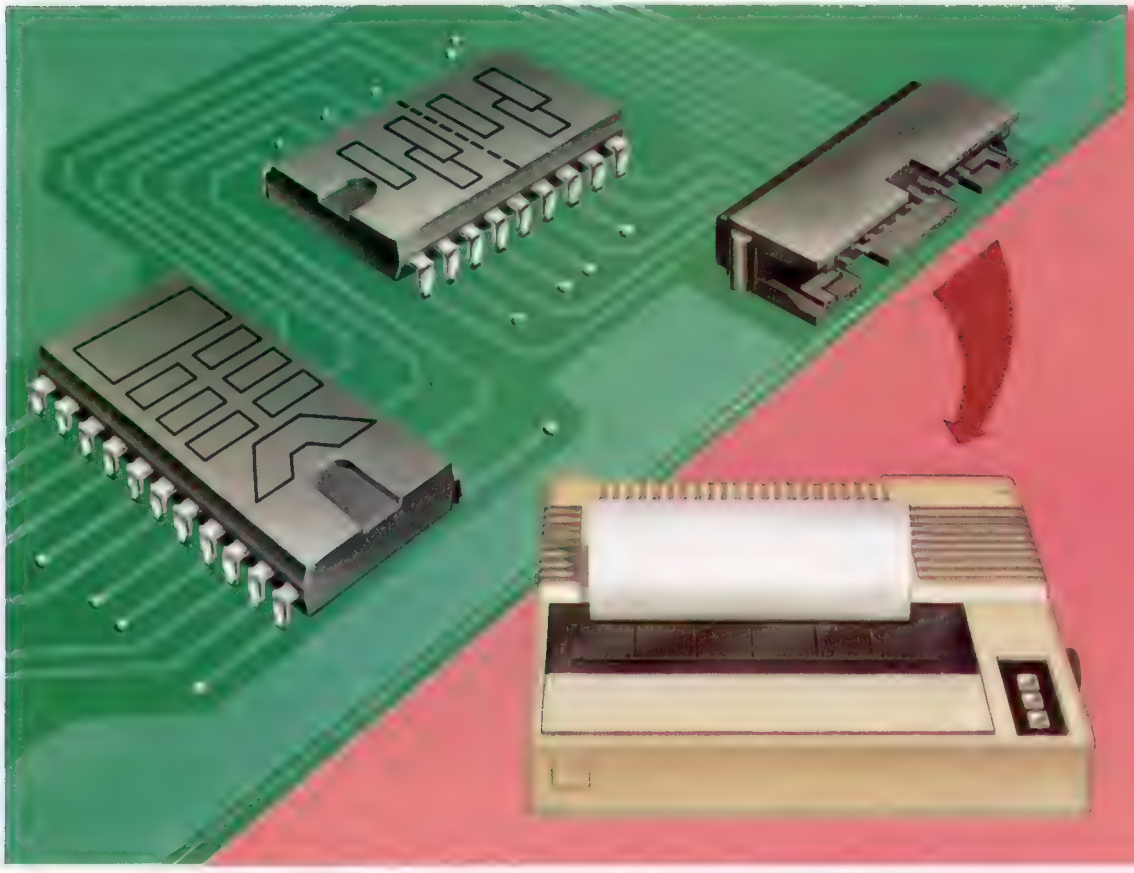
Certains vendeurs de matériel de photocomposition et imprimeurs placent eux-mêmes chez leurs clients des consoles comprenant clavier et écran, permettant une saisie simple et complète des documents à photocomposer. La société MicroCompo, en Belgique, se range par exemple dans cette catégorie : elle loue à ses clients des consoles appelées « CompoType », pour des travaux ponctuels. Une imprimante de contrôle édite un listage reprenant tous les signes spéciaux utilisés en imprimerie. L'écran graphique permet de monter une page beaucoup plus rapidement que sur une table de montage.

Le principal avantage de ce procédé de « saisie décentralisée » est d'éviter une deuxième saisie de textes déjà tapés sur un matériel de traitement de texte, avec toutes les erreurs supplémentaires que cette nouvelle saisie risque d'entraîner.

Certes, s'il n'y a pas une deuxième saisie du texte, la dactylo devra pratiquement être transformée en claviste : le travail autrefois effectué par le linotypiste est à présent transféré sur l'entreprise utilisatrice, voire sur l'auteur du texte lui-même. Toutefois, selon MicroCompo, l'éditeur de texte fourni par cette société, « CompoText », n'exigerait de la part de la dactylo qu'un quart d'heure d'apprentissage pour effectuer une saisie sur le « CompoType ».

Entrées et sorties

L'un des plus importants aspects de la programmation en langage d'assemblage est le contrôle des entrées et sorties. Nous allons voir le fonctionnement des deux puces 6820 PIA et 6850 ACIA.



Affaires de périphériques

Les imprimantes ont besoin qu'on leur envoie des données dans des formats particuliers et à certaines vitesses : il serait superflu de confier des tâches aussi triviales à l'UC; c'est pourquoi l'UC envoie les caractères de données à l'adaptateur d'interface périphérique (PIA) et se consacre à plein temps à la communication avec l'imprimante.
(Cl. Kevin Jones.)

Le processeur 6809, comme le 6502, mais contrairement au Z80, ne possède pas un espace d'adresses séparé pour les entrées/sorties, d'instructions particulières d'E/S. Les puces d'interface d'E/S sont logées dans l'espace adresse normal et sont traitées en utilisant les instructions d'accès à la mémoire. Pour le processeur, ces dispositifs apparaissent comme des emplacements mémoire.

Certains dispositifs d'entrées/sorties peuvent être liés directement au bus de données du système; mais, normalement, il existe une puce d'interface entre les deux. Ces puces d'interface sont des dispositifs sophistiqués, aussi complexes que le microprocesseur lui-même. Il est normal d'utiliser des puces appartenant à la même famille que le processeur, puisque cela facilite la liaison et le contrôle. Les deux puces les plus usitées avec le 6809 sont le 6802 (ou 6821) PIA (adaptateur d'interface périphérique), qui traite les entrées/sorties parallèles, et le 6850 ACIA (adaptateur d'interface de communication asynchrone), qui traite les entrées/sorties séries. Chacune d'elles comporte plusieurs registres, et leur contrôle consiste à en lire et à en écrire le contenu, en les traitant comme s'il s'agissait d'emplacements mémoire normaux. Il y a trois types de registres :

- *Registres de contrôle :* ce sont des registres où l'on ne peut qu'écrire; les valeurs y sont stockées afin de programmer la puce pour les options particulières dont vous avez besoin, telles que la détermination de la vitesse en bauds.
- *Registres d'état :* ce sont des registres où l'on ne peut que lire; leurs valeurs donnent une indication de l' « état » de la puce. Ces registres montreront, par exemple, si une entrée a été reçue, si la dernière sortie a été transmise, ou si une erreur est survenue.
- *Registres de données :* ce sont les registres qui contiennent les données en entrée ou sortie, de sorte qu'elles puissent être lues et/ou écrites.

Afin de conserver de l'espace mémoire, il est courant que plus d'un registre occupe la même adresse. Par exemple, un registre d'état et un registre de contrôle peuvent être à la même

adresse; celui qui apparaît à cette adresse est déterminé par ce que vous désirez y lire ou y écrire. De même, un registre d'entrée de données et un registre de sortie de données peuvent partager une adresse.

Le 6820 PIA contient six registres et occupe 4 octets de mémoire consécutifs. La puce comporte en fait deux ports indépendants, dont chacun utilise trois registres. Le côté périphérique de la puce a 8 lignes de données et 2 lignes de contrôle pour chaque port. Les 2 lignes de contrôle seront connectées aux lignes de contrôle appropriées sur le périphérique, de sorte qu'elles puissent servir à déterminer l'état. La ligne de contrôle 1 est seulement pour les signaux de contrôle d'entrée, mais la ligne de contrôle 2 peut être programmée pour recevoir ou émettre des signaux de contrôle.

Voici les trois registres :

- *Registre de données :* peut fonctionner à la fois pour les entrées et les sorties, puisque chaque bit peut être mis indépendamment.
- *Registre de direction de données :* chaque bit sert à mettre le bit correspondant dans le registre de données en entrée (0) ou sortie (1).
- *Registre combiné contrôle/état.*

Le registre de direction de données et le registre de données partagent la même adresse. L'état de l'un des bits du registre de contrôle détermine lequel apparaît à cette adresse. Le tableau dans la marge donne le décalé de l'adresse de base de la puce pour les adresses de chacun des registres.

Registre	Décalé
Donnée A	0
Direction de donnée A	0
Contrôle/État A	1
Donnée B	2
Direction de donnée B	2
Contrôle/État B	3

Les bits dans le registre contrôle/état sont assignés comme suit :

Bit	Fonction
7	Bit d'état pour ligne de contrôle 1; mis à 1 lorsqu'un signal de contrôle est reçu et automatiquement remis à zéro lorsque le registre de données est lu.
6	Bit d'état pour ligne de contrôle 2; comme bit 7.
5	Détermine si ligne de contrôle 2 est utilisée pour l'entrée (0) ou la sortie (1).
4	Détermine la nature du signal de contrôle sur ligne 2.
3	Si ligne de contrôle 2 est mise pour entrée, alors un 1 permet des interruptions à partir du bit 6; si pour sortie, alors cela permet de déterminer la nature du signal.
2	Sélectionne données (1) et direction de données (0).
1	Détermine la nature du signal de contrôle sur ligne 1.
0	Un 1 ici permet des interruptions à partir du bit 7.

Pour le moment, nous ne considérerons pas l'utilisation des interruptions, ni les divers effets des bits 1 et 4. Notez qu'en écrivant dans le registre pour mettre les bits de contrôle, il est impossible d'affecter les bits 6 et 7.

Le premier de nos programmes utilise une puce 6820 pour contrôler une imprimante via une interface standard Centronics. Celle-ci spécifie un grand nombre de lignes de contrôle ainsi que les 8 lignes de données. Nous ne nous occupons pas de leurs détails, sauf pour noter qu'une ligne de contrôle (appelée *échantillon*) sert à signaler à l'imprimante qu'un caractère est en route. Elle doit être connectée à la ligne de contrôle 2, qui doit être mise pour la sortie. Un autre signal de contrôle (appelé *accusé de réception*) sert à l'imprimante à indiquer qu'elle est prête pour le prochain caractère à émettre. Elle doit être connectée à la ligne de contrôle 1. Les 8 lignes de données doivent être clairement connectées aux 8 sorties de données à partir du port PIA.

Pour mettre le port, il nous faut sélectionner le registre direction de données et programmer tous les 8 bits pour la sortie, puis sélectionner le registre données et mettre la ligne de contrôle 2 pour la sortie. Pour utiliser la puce, nous lisons constamment le registre de contrôle/état, jusqu'à ce qu'un 1 apparaisse dans le bit 7, indiquant que l'imprimante est prête pour un caractère. Nous pouvons alors écrire un caractère au registre de données, qui envoie automatiquement un signal de contrôle sortie sur la ligne de contrôle 2. Le bit 6 sera mis à 1 lorsque le caractère aura été transmis. Nous devons alors lire le registre de données pour effacer les bits 6 et 7, et répéter le processus, jusqu'à ce que le dernier caractère ait été transmis. Le processus d'émission et de réception de signaux de contrôle entre le processeur et le périphérique est appelé *handshaking* (« poignée de mains »).

Nous supposerons que l'adresse de base du PIA est donnée dans une table d'adresses placée en $3000. A l'entrée en sous-programme d'impression, le registre de processeur A contient l'index dans cette table, et Y contient l'adresse de la chaîne à imprimer.

La chaîne est stockée dans le format normal, c'est-à-dire d'abord l'octet de longueur. Il y a deux sous-programmes, l'un pour mettre le port et l'autre pour imprimer la chaîne.

6850 ACIA

Le 6850 ACIA est un UART (récepteur/transmetteur asynchrone universel) qui sert pour la communication série, utilisant normalement le protocole RS232 et éventuellement un modem. Il a quatre registres et occupe deux adresses. Il y a cinq connexions de la puce côté périphérique : une ligne pour les données transmises, une pour les données reçues, et trois lignes de contrôle pour la poignée de mains, si nécessaire. Deux d'entre elles sont pour l'entrée de signaux de contrôle-DCD (détecte porteur de données) et CTC (prêt à envoyer), et l'un pour les signaux en sortie — RTS (demande d'émission). L'emploi de ces lignes devrait être assez évident d'après leur dénomination, et elles peuvent être connectées aux lignes du même nom sur un RS232 standard.

Registre	Décalé
Registre contrôle	0
Registre d'état	0
Donnée transmise	1
Donnée reçue	2

Les quatre registres ACIA sont donnés en marge. Dans le registre de contrôle, le bit le plus significatif (bit 7) sert à permettre des interruptions pour recevoir des données. Les bits 5 et 6 servent à valider ou invalider les interruptions de transmission et à déterminer la nature du signal de contrôle émis sur la ligne RTS. Les bits 2, 3 et 4 servent à déterminer la taille du bloc logique qui est transmis. Lorsqu'un octet est transmis sur une liaison série, il y a généralement au moins 10 bits émis, commençant par un bit de début,

qui est détecté par le récepteur afin qu'il sache qu'une donnée suit. La donnée elle-même peut avoir 7 ou 8 bits, et peut comporter 1 bit de parité. Enfin, il peut y avoir 1 ou 2 bits d'arrêt. Voici les différentes options disponibles :

Registre de contrôle			Nombre de bits de données	Parité	Nombre de bits d'arrêt
Bit 4	Bit 3	Bit 2			
0	0	0	7	Paire	2
0	0	1	7	Impaire	2
0	1	0	7	Paire	1
0	1	1	7	Impaire	1
1	0	0	8	Néant	2
1	0	1	8	Néant	1
1	1	0	8	Paire	1
1	1	1	8	Impaire	1

Les deux derniers bits significatifs (0 et 1) servent à déterminer la vitesse de transmission et de réception. Cela est fait en mettant un diviseur pour la fréquence d'horloge. Le 6850 n'a pas sa propre horloge et on doit donc lui en fournir une externe, qui est spécifiquement à 1,760 Hz.

Registre de contrôle		Diviseur de fréquence d'horloge
Bit 1	Bit 0	
0	0	1
0	1	16
1	0	64

La combinaison qui n'est pas montrée dans la table, où les deux bits sont à 1, a pour effet une réinitialisation principale de la puce.

Dans le registre d'état, les bits ont les fonctions suivantes :

Bit	Fonction
7	Demande d'interruption.
6	Mis si une erreur de parité a lieu à réception.
5	Mis en cas d'engorgement du récepteur, c'est-à-dire si trop de bits ont été reçus, avec superposition des caractères.
4	Mis si une erreur de synchronisation se produit à la réception — le mauvais nombre de bits de début et d'arrêt.
3	Mis lorsqu'un signal est reçu sur la ligne CTS.
2	Mis lorsqu'un signal est reçu sur la ligne DCD.
1	Mis lorsqu'un registre de données transmises est vide.
0	Mis lorsqu'un registre de données reçues est plein.

Notre second programme utilise une puce 6850 pour recevoir une chaîne de caractères, terminée par un retour de chariot, à partir d'un terminal éloigné. Le principe consiste à programmer convenablement la puce, puis à faire une boucle pour vérifier si le registre de données reçues est plein. Dans ce cas, nous enlevons l'octet de donnée, qui réinitialise le bit 0 dans le registre d'état. Le processus est réitéré jusqu'à ce que le caractère reçu soit un retour de chariot (code ASCII 13). Nous ignorerons toute erreur de transmission, bien que la vérification de ces erreurs soit très facile. Nous supposerons par convention : 8 bits de données, pas de parité et 2 bits d'arrêt, et une vitesse d'horloge divisée par 16. Le premier sous-programme programme la puce, le second reçoit les données.

Programme PIA

```
TABLE   EQU   $3000          Sous-programme pour valider
        ORG   $1000          le port A.
        ASLA                 Décale A à gauche pour
                             multiplier par 2 (la table
                             consiste en adresses à
                             2 octets).
        LDX   #TABLE         Donne l'adresse de base
        LDX   A,X            de PIA.
        CLR   1,X            Donne l'accès au registre
                             de direction de données.
        LDB   #%11111111     Met tous les bits pour sortie.
        STB   ,X             Invalide interruptions, met
        LDB   #%00101100     ligne de contrôle 2 pour
        STB   1,X            sortie et sélectionne
                             registre de données.
        RTS
                             Sous-programme pour imprimer
                             une chaîne dont l'adresse est en Y.
        ASLA                 Décale A à gauche pour
        LDX   #TABLE         multiplier par 2.
        LDX   A,X            Donne l'adresse de base de PIA
        LDA   ,Y+            Donne la longueur de
                             chaîne dans A.
LOOP1   BEQ   FINISH         Vérifie si la longueur est nulle
LOOP2   LDB   1,X            Vérifie si prêt pour bit suivant
        ANDB  #%10000000     Masque tout sauf le bit 7.
        BEQ   LOOP2          Si pas prêt.
        LDB   ,Y+            Prend caractère suivant.
        STB   ,X             L'imprime.
LOOP3   LDB   1,X            Vérifie s'il est transmis.
        ANDB  #%01000000     Regarde bit 6.
        BEQ   LOOP3          Boucle si pas encore prêt.
        LDB   ,X             Lit registre de données
                             pour effacer les bits d'état.
        DECA                 Soustrait 1 de la longueur.
        BRA   LOOP1          Prend caractère suivant.
FINISH  RTS
```

Programme ACIA

```
TABLE   EQU   $3000          Sous-programme pour
        ORG   $1000          programme 6850.
                             Sous-programme pour valider
                             ACIA.
ACIAST  ASLA                 Décale A à gauche pour
                             multiplier par 2 (table
                             d'adresses à 2 octets).
        LDX   #TABLE         Donne adresse de base de
        LDX   A,X            ACIA.
        LDA   #%00000011     Réinitialisation maître de
        STA   ,X             ACIA.
        LDA   #%00010001     Dans le registre contrôle.
        STA   ,X             Programme ACIA (8 bits de
        RTS                  données, pas de parité,
                             2 bits d'arrêt).
                             Sous-programme pour accepter
                             des chaînes de caractères.
BUFFER  EQU   $4000          Endroit où mettre la chaîne.
CR      EQU   13             Code ASCII retour chariot.
        BSR   ACIAST         Valider ACIA. Le registre X
                             contient l'adresse ACIA.
        LDY   #BUFFER        Destination dans Y.
LOOP    LDB   ,X             Met état.
        ASLB                 Décale bit 7 hors du
                             registre B dans le drapeau
        BCC   LOOP           de retenue du CCR.
        LDA   1,X            Retour si ce bit n'était pas
        STA   Y+             mis, c'est-à-dire pas
        CMPA  #CR            d'interruption.
        BNE   LOOP           Donne octet de donnée.
        RTS                  Le stocke.
                             Y a-t-il retour chariot ?
                             Caractère suivant.
```

Page manquante
(publicité)

Page manquante
(publicité)